(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 713 174 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
22.05.1996 Bulletin 1996/21

(51) Int Cl.6: **G06F 9/44**, A63F 9/22

(21) Application number: 95307727.8

(22) Date of filing: 30.10.1995

(84) Designated Contracting States:
DE FR GB NL SE

(30) Priority: 31.10.1994 US 332813

(71) Applicant: NINTENDO CO., LTD.
Kyoto-shi, Kyoto-fu (JP)

(72) Inventors:
- Yamato, Satoshi, c/o Nintendo Co, Ltd
Kyoto-shi, Kyoto-fu (JP)
- Suzuki, Toshiaki, c/o Nintendo Co, Ltd
Kyoto-shi, Kyoto-fu (JP)
- Hotta, Takuji, c/o Nintendo Co, Ltd
Kyoto-shi, Kyoto-fu (JP)
- Nishiumi, Satoshi, c/o Nintendo Co, Ltd
Kyoto-shi, Kyoto-fu (JP)
- Fujiwara, Kazuhiro, c/o Nintendo Co, Ltd
Kyoto-shi, Kyoto-fu (JP)

(74) Representative: Stevens, Ian Edward
Stevens, Hewlett & Perkins,
1 St. Augustine's Place
Bristol BS1 4UD (GB)

(54) Video game/videographics program fabricating system and method with superimpose control

(57) A videographics/video game fabricating system includes a multiprocessor based game processor console which includes a main central processing unit (CPU) which controls editing operations and operating system task execution and a game CPU for executing the model video game which is loaded into a pluggable RAM cartridge. The model video game provides a starting point from which a user can readily create an original video game including desired aspects of the model software. The system permits a user to modify any of the game's moving objects. background screens. music or sound effects. The main CPU and game CPU cooperate in the game execution and editorial process such that an editing screen generated by the main CPU is superimposed on a game screen generated by the program executing CPU. The game processing console includes ports for interconnection with a wide variety of peripheral devices including a standard television set. keyboard. game hand controllers. mouse. modem board. an interface board for coupling the game processor to a personal computer system. floppy disk drive. an external RAM game cartridge and a user"s ID card. The system utilizes unique "unit" based data structures in which moving objects are processed on a unit basis and where each object is assigned a unit ID which is associated with a wide range of object. game characteristics. game processing and location data including status information. present screen display location. object format. character size. pose information. collision threshold information. tempo information. attribute data. animation data together with address pointers identifying other processing related information associated with the identified object. A wide range of information is likewise stored in data structures associated with background screens referred to as "stage" data.

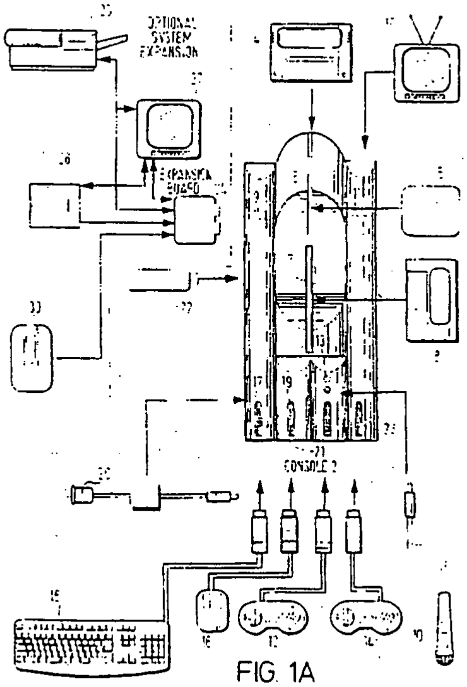


FIG. 1A

Printed by Jouve 75001 PARIS (FR)

game play related characteristics. Once the moving object is selected. further icons are displayed permitting the user to completely change the object"s character dot pattern for one or more of the poses associated with the object. animation features related to the object. the responses associated with detected game play conditions associated with the object. the collection of statuses associated with each object. the pattern of the object"s movement. the sound associated with the pattern of the objects movement and a wide variety of additional game play related characteristics. The screen background may be likewise modified by accessing a stage window permitting the entire background map. the music associated with the background and a wide variety of additional background related features to be edited.

The exemplary embodiment of the present invention uses a multiprocessor based game processor console which includes a main central processing unit (CPU) controlling editing operations and operating system task execution and a game CPU for executing the model video game that is loaded into a pluggable RAM cartridge. The model video game provides a starting point from which a user can readily create an original video game using desired aspects of the model game. The model video game can be readily modified to such an extent it appears to be a completely new game. The system permits a user to modify any of the game"s moving objects. background screens. music or sound effects.

The main CPU and game CPU cooperate in the game execution and editorial process such that an editing screen generated by the main CPU is superimposed on a game screen generated by the program executing CPU. The game processing console includes ports connected to a wide variety of peripheral devices including a standard television set. keyboard. game hand controllers. mouse. modem board. an interface board for coupling the game processor to a personal computer system. floppy disk drive. an external RAM game cartridge and a user"s ID card.

The system utilizes unique "unit" based data structures in which moving objects are processed on a unit basis and where each object is assigned a unit ID which is associated with a wide range of object. game characteristics. game processing and location data including status information. present screen display location. object format. character size. pose information. collision threshold information. tempo information. attribute data. animation data together with address pointers identifying other processing related information associated with the identified object. A wide range of information is likewise stored in data structures associated with background screens referred to herein as "stage" data. Programming is structured for ease of user editing using condition and process related operation tables and unit pointers that identify object unit data structures which are to be processed. Unit operation tables are utilized for processing the units and identify both a predetermined condition and the processing operation to be performed upon detection of the predetermined condition for each unit. Both condition and process operation may be changed by the user by modifying these tables.

These and other objects. features. aspects and advantages of the present invention will become more apparent from the following detailed description of the illustrative embodiment of the present invention. when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1A is a block diagram of a videographics game fabricating system and FIGURE 1B is a perspective view of the game processor system console unit 2 shown in FIGURE 1A:

FIGURES 2A and 2B are a more detailed block diagram of an illustrative embodiment of the game processor system shown in Figure 1A:

FIGURES 3A and 3B are memory maps of a part of the system memory space:

FIGURE 4 is an exemplary on-line networking system in which the illustrative embodiment of FIGURES 2A and 2B may be used:

FIGURE 5 is an exemplary title screen for accessing the exemplary program authoring tools of the illustrative embodiment:

FIGURE 6 is an illustrative system break screen:

FIGURE 7 depicts an exemplary character editing screen:

FIGURE 8 is an exemplary screen accessed when a user clicks on the character icon:

FIGURE 9 shows an illustrative "all" status screen display for the unit named "Mario":

FIGURE 9A is an illustrative key editing screen:

FIGURE 10 is an illustrative unit animation editing screen:

FIGURE 11 is an illustrative background screen editor:

FIGURE 12 shows an illustrative "map editor":

FIGURE 13 illustrates how the background of the game display screen may be modified by dragging a background screen and pasting it onto an active background:

FIGURE 14 shows an illustrative music editing screen:

FIGURE 15 is an illustrative music editor screen which permits music attributes to be changed for each sound course:

FIGURE 15A is an illustrative music editor screen which permits music to be changed in real time while the music is being played back:

FIGURE 16 is an illustrative screen display depicting the operation of the auto programmer:

FIGURE 17 is an exemplary status editor display screen:

FIGURE 18 is an exemplary "stage" window display screen which permits a vast range of background

The game processor system console unit 2 includes an insertion port 5 for a game processor ID card 6. In an exemplary embodiment of the present invention. the ID card includes a security code which is compared against data stored at a predetermined location on floppy disk 8. If the comparison results in a determination of authenticity. then data from floppy disk 8 may be successfully transferred to RAM cassette 4. Although floppy disk 8 contents may be copied. the user is only issued one ID card 6 to thereby provide a measure of security against counterfeiters. The ID card 6 may also contain the user"s photograph and/or other identification data.

The game processor system console unit 2 shown in Figure 1A is designed to be coupled to a wide variety of input/output devices. It includes a jack for microphone 10 connection to enable the input of sound signals that may be used during game play. Additionally. the game processor system console 2 includes at least two player controllers 12 and 14. which may be of the type described in U.S. Patent No. 5.207.426. These controllers are utilized to control video game play on a user"s television screen by controlling moving object character movement and associated special effects in. for example. the manner typically utilized in commercial available SNES controllers described in the above-identified patent.

The game processor system console 2 additionally includes ports for connecting a conventional mouse 16 and PC compatible keyboard 18. The mouse 16 and keyboard 18 are utilized in a manner which will be described in detail below by the user as graphic/user interfaces during the video game design process. The mouse 16 permits a user such as an elementary school child who is totally unfamiliar with game programming techniques to create a unique video game through the use of the icon driven system described herein. The keyboard 18 permits game modification through input. for example. of game software instructions by more sophisticated users such as game programmers.

The game processor system unit console 2 also includes a connection port for a modem board 22. By way of example only modem 22 is a 9600 baud. half-duplex modem. The modem 22 permits the game processor system to be used in an on-line network described below. The system also includes an A/C adaptor 20 for providing power to the system.

The game processor system unit console 2. as will be described further below. includes two central processing units. One is primarily responsible for video game play control and a second is primarily responsible for game editing related tasks and for executing the operating system program for controlling the transfer of information from the game processor disk 8 to RAM cassette 4.

The system components shown in Figure 1A and described above permit a user who is totally unfamiliar with video game program development to create a wide range of video games using model software stored on floppy disk 8. Through the use of the optional components identified in Figure 1A. the system may be expanded to more readily permit professional game program designers to create video games in a unique emulation system. In this alternative embodiment. the game processor system includes an expansion board 24 which couples further I/O and other components to. for example. the operating system CPU within console 2. As shown in Figure 1A. various additional I/O components may be coupled to the system such as a scanner 30. hard disk drive 28 or printer 26. Scanner 30 may be a conventional optical scanner and is utilized to scan a graphical image. digitize the image for storage in the game processor system unit"s console memory system for use in a video game being designed. A user would then be able to access the stored image. add colors and modify the image. An SCSI interface may be embodied on expansion board 24 to permit coupling to an IBM compatible PC 27.

Figure 1B is a perspective view of the game processor system console unit 2. The console 2 includes a power ON/OFF switch 11 and a reset button 9. The reset switch 9 permits resetting the entire system including the operating system executing CPU and the game CPU. The reset button 9 in addition to placing the game program executing CPU at a known initial state also serves to interrupt the operating system CPU to permit. for example. testing operations to be performed. As shown in Figures 1A and 1B receptacles 5 and 7 are slots for receiving the game processor ID card 6 and the floppy disk 8. respectively. Both receptacles 5 and 7 have associated recessed areas to permit a user to easily grab and extract the respective ID card 6 or floppy disk 8. The floppy disk receiving mechanism is further described in the above identified copending application. Serial No. 08/332.812 filed October 31. 1994. which application has been incorporated herein by reference. As shown in Figure 1B. the console unit also includes a floppy disk eject button 3. Additionally. as shown in Figures 1A and 1 B. connectors 13. 17. 19. 21 and 23 are exposed to permit ready connection of microphone 10. keyboard 18. mouse 16. controller 12 and controller 14. respectively.

Figures 2A and 2B are a more detailed block diagram of the game processor system shown in Figure 1A and are an illustrative embodiment which specifies details such as the number of connector pins. particular types of memory devices. etc. These and other implementation details are set forth only as an illustrative arrangement of one of many possible alternative arrangements for implementing the invention and should not be construed as limiting the scope of the present invention.

Figures 2A and 2B identically numerically label many of the same input/output devices shown in Figure 1A and show connections between the I/O devices and game processor system console 2 in greater detail than Figure 1A. The controllers 12. 14. keyboard 18 and mouse 16 are coupled to the game processor system

The game processor system console 2 also includes an operating system or main CPU 228 which may be. for example. a NEC V810 processor which is a 32 bit RISC processor. Alternatively. for example. a Motorola 68000 Series Processor or other processors having similar processing power may be utilized. The main CPU 228 processes operating system related routines in parallel with the game CPU 200 which executes a video game program. as will be described further below. Main CPU 228 and game CPU 200 communicate through gate array 226 which is described in detail in conjunction with Figure 23. As indicated above. processor 228 (like game processor 200) includes an associated picture processing unit 224 for performing graphic processing operations to reduce the graphics processing burden on CPU 228. The operating system CPU 228 is coupled directly to video RAM 220 or indirectly via picture processing unit 224 depending upon the control setting of bus selector 222. As noted above. PPU"s 212 and 224 are of the type described in the above identified SNES related patents and patent applications which have been incorporated herein by reference.

In accordance with the presently preferred embodiment of the present invention. main CPU 228 is a processor having a 32 bit bus width and operates at a 21.477 MHz clock rate . The CPU 228 is coupled to. for example. a 4 megabyte DRAM 230. The work DRAM 230 is expandable. if desired. to up to 24 megabytes via an extension DRAM 232. Main CPU 224 is additionally coupled to. for example. an 8 mega bit ROM 234 which stores an initial program loader (IPL) subroutine a BIOS operating system program. and character fonts to be accessed by CPU 228. CPU 228 also has access via its system bus to an 8 mega bit flash memory 236 which is utilized. for example. for software backup for data from floppy disk drive 199. The main CPU 228 system bus is also coupled to. for example. a 10 bit A/D converter 238 (which is utilized to digitize sounds coming from. for example. microphone 10 via board 197). a floppy disk controller 240 and real time clock 242.

The main CPU 228 and the game CPU 200 boot up independently. After power is turned on. the game CPU 200 is initially placed in a reset state and then restarted. Initially. the game CPU 200 executes program instructions out of monitor ROM 204. The main CPU 228 controls the areas of memory which are accessed by game CPU 200. The main CPU 228 loads a register in gate array 226 which identifies whether CPU 200 is to execute out of monitor ROM 204 or RAM cartridge 4. The main CPU 228 executes the initial program loading and operating system instructions in ROM 234 to read out the contents of floppy disk 8 including the model game software via floppy disk drive 199 and floppy disk controller 240. Operating system program instructions for controlling operations associated with the model game software are transferred to DRAM 230 for execution by main CPU 228. After system power has been turned off and thereafter turned on. main CPU 228 checks flash memory 236. Upon being first turned on. operating system programs are read out from floppy disk 8 and stored in flash memory 236 and transferred to DRAM 230. Upon being turned on for the second time. operating system programs are transferred from flash memory 236 to DRAM 230 without use of the floppy disk 8. In general. the flash memory"s access speed is faster than the floppy disk"s access speed. Therefore. the system can be quickly started at the second turn on.

Model software information from floppy disk controller 240 is initially buffered in DRAM 230 for transfer to the RAM cartridge 4. Main CPU 228 then generates a communication ready signal and transmits the signal to a handshake port in gate array 226. Game CPU 200. through instructions stored in monitor ROM 204. monitors the handshake port and prepares for reception of data and/or instructions from the communication RAM in gate array 226. The handshake port may be illustratively implemented by a pair of one way handshake ports. A one way handshake port communicates information from the Main CPU 228 to Game CPU 200. Another one way handshake port communicates the information from the Game CPU 200 to Main CPU 228.

The handshake port includes a buffer register which is addressable by game CPU 200 which indicates whether information is to be transferred between main CPU 228 and game CPU 200. When game CPU 200 has determined that a communication RAM embodied within gate array 226 has received information from main CPU 228. game CPU 200 accesses the information from the communication RAM and transfers such information to RAM cartridge 4 via 80 pin connector 189 and connector board 186. The main CPU 228. after the communication RAM information has been transferred to RAM cartridge 4. sets a register embodied within gate array 226 which initiates switching of control of game CPU 200 from monitor ROM 204 to RAM cartridge 4. After the game CPU control has been switched to RAM cartridge 4. game CPU 200 no longer monitors the handshake port or has any further interaction with the communication RAM within the gate array. At this point. game CPU 200 operates to execute game related instructions out of RAM cartridge 4 in a manner substantially the same as the applicants game machine described in the above identified SNES related patents and patent applications.

The main CPU 228 constantly monitors information transmitted via the CPU 200 system bus by accessing a register copy RAM embodied within gate array 226. The information written into the register copy RAM from the game CPU 200 bus is formatted such that it"s compatible with the main CPU 228 format. Information from PPU 212 and sound processor 208 flows into the register copy RAM for monitoring by main CPU 228 such that main CPU 228 is continuously aware of the game state. The use of a 32 bit RISC processor provides main CPU 228 with the processing power necessary to monitor such a large volume of information. As is described be-played. After a user selects whether the game is to be played by one or two players. a user depresses the start button of controller 12 or 14 shown in Figure 1A. The game specified by the model software. e.g.. a version of the well known "Mario Bros." game. then begins play.

At any point in the game in which the user desires to change any displayed moving object character or any portion of the display screen background. the user stops the game to initiate what is referred to herein as a "system break". When a system break is initiated. as shown in Figure 6. the system break screen is displayed in which a system window screen 50 is superimposed over a user selected game screen 57. A moving object character. such as the well known "Mario" prominent in video games manufactured by the applicant . or any portion of the background may be modified by clicking on the area where a change is to be made.

The present invention permits wide ranging changes to the model software such that the model software game background and moving objects are modified to the extent that the original model software game is unrecognizable. By way of example only. Figure 6 illustrates that a change is to be made to the moving object character recognizable as "Mario" by the user "clicking" the mouse when the cursor (e.g.. the hand) overlays the character to be modified. If desired. a completely new character may be substituted for Mario.

For the purposes of this example. presume that a user playing the game wishes to remove Mario"s moustache. As noted above. a user stops the game by initiating a system break upon clicking the mouse. Thereafter. the user clicks on Mario. Main CPU 228 generates. for example. a rectangular block circumscribing the character to be changed together with an indication 52 of the character"s "unit number". Each of the moving objects and background portions of the game which may be modified are identified by unique unit numbers. Associated with each unit number and its object. as will be described in detail below. are a wide range of characteristics such as. for example. size. poses. animation paths. etc. The system window 50 includes a series of icons which permit the user to select a wide range of editorial functions. The system window "stage" icon 54 enables access of the "stage" window shown in Figure 18 which permits a vast range of background character editing. The system window also includes a return to the title screen icon 55. a reopen game icon 59. and a switchable control 53 which permits the user to switch between a game stop and slow play mode. Numerous additional or alternative editing functions could be included in the system window 50. if desired in the system window 50 to identify particular game fabrication/editing tools which may be selected.

As can be seen in Figure 6. system window 50 and its editing tool indicating icons as well as unit number indication 52 are superimposed on game screen 57. The selection of the Mario character for editing results in the display of a sequence of Mario editing screens/windows Other moving objects may be selected for editing in a similar manner.

Figure 7 depicts an exemplary character editing screen. It shows a Mario character editing related window which visually indicates that it is the Mario character being modified (60). Below the indication of Mario. the current status of Mario is identified (62). The status of "Mario" is indicated as being "stopped". Other exemplary status indications which could have been identified are Mario "moving left". a Mario "moving right". a Mario "jump" indication or a Mario appearing from the top or bottom. By changing the status of Mario. which may be done using the status editors described below. the game is changed accordingly.

Below the status block. a sequence of icons are shown indicating exemplary Mario related editorial modifications which may be made and editing screens which may be accessed by clicking on the related icon. Icon 64 enables changes to be made with respect to the Mario animation patterns as described below in conjunction with Figure 10. Icon 66 permits editing of character "status" conditions as described below in conjunction with Figure 17. Icon 68 permits changes to be made to the special sound effects associated with the video game when Mario is moved. Icon 70 resumes game play. Icon 72 permits changes to be made to the "all status" conditions as described below in conjunction with Figure 9. Icon 74 enables changes to be made with respect to the Mario character dot pattern as described below in conjunction with Figure 8. Icon 76 permits editing of character movement as described below in conjunction with Figure 16. Icon 78 causes the system to return to the system window.

For example. to remove Mario"s mustache. a user clicks on character icon 74 to generate the display screen shown in Figure 8. By clicking on the character icon. control is transferred to the character design tool shown in Figure 8. A zoom tool may be selected by clicking on icon 85 which results in an expanded view of the character to be modified. A pencil tool may be selected by clicking on icon 86 which switches to pencil mode/box mode. Pencil mode permits changes of the graphic data on a dot by dot basis. Box mode permits the graphic data to be painted by any square area. Icon 84 controls resuming game play or returning to the system break screen. An erase tool may be selected by clicking on icon 87 which erases all graphic data on a pattern of. for example. Mario. An undo tool may be selected by clicking on icon 88 which cancels the immediately preceding command. As shown in Figure 8. a color palette 80 is displayed to permit a user to select any of a number of colors to change the color of any portion of Mario by clicking on a selected color and drawing on Mario with a coloring tool (e.g.. crayon shaped cursor) in accordance with conventional painting programs. This color palette function enables a user to select colors within a given color palette. select between various palettes of color and create new colors with a color spectrum func-

and the animation sequence tempo are controlled based on the settings of Figure 10.

Poses 8 through 15 shown in Figure 10 (collectively 108) identify alternative poses which may be inserted in the animation sequence. A user need only click on a particular pose. e.g., 10. drag the pose and place it over. for example. animation sequence pose 2. This will result in pose 10 being displayed in place of pose 2.

The area defined by four corner marks shown in the upper lefthand corner of Figure 10 represents the collision threshold 109 for the particular character. i.e.. during game play a collision will be detected if the identified character appears within the distance indicated by the corner mark area to another object. The corner mark area size may be controlled by the "HIT" indicator control threshold control 112.

Turning back to Figure 6. if. after a system break has been initiated. the user clicks on the stage icon 54 on system window 50. the stage window shown in Figure 18 is displayed on the screen. A "screen editor" is accessed such as shown in Figure 11 by clicking on the screen icon 152 on the stage window. Using the screen editor. if desired. a background element from the bottom portion of Figure 11 may be dragged up to the background screen portion to modify the actual display background. The screen editor permits the user to review and modify any one of a number of different background layers associated with the game at the point in time of the system break. The screen editor includes controls for horizontally and vertically scrolling the background to permit the user to view the entire background through a user accessible "view" mode which permits selection of. for example. a first. second or third background screen. Through the screen editor each of the different background layers are accessible and scrolling may be controlled to view the entirety of each of the background layers as well as to modify each layer.

The screen editor also has an associated "map editor" which may be opened such as that shown in Figure 12. The map editor is available to relocate available background screens which are shown in a reduced form in the map editor. Using the map editor "pasting" function. the background of the game display screen may be modified by. for example. dragging a background screen drawing a character such as indicated at 120 in Figure 13 and pasting it onto an active background.

The music associated with the video game background may be modified by accessing the music editor by clicking on a musical symbol icon associated with a "STAGE" window. which will be described further below. to result in a screen display such as shown in Figure 14. As indicated in Figure 14. the music editor displays a game music sheet. The "switches" (which. during music edit. are displayed in color) correspond to similarly colored figures on the game music sheet display. By selecting. via the mouse. one of the displayed switches shown in Figure 14. it is possible to playback only the bass portion of the music. the piano portion of the music or any other instrument that is selected. By selectively layering the bass. piano and other musical sounds. the background music is formulated.

If a colored symbol is clicked in Figure 14 identifying a selected musical segment. the window shown in Figure 15 is generated and overlayed on the musical editor screen of Figure 14 to thereby permit individual control over musical tone, sound volume. etc. for each selected numerical segment. For example: if the "KEY" control shown in Figure 15 is manipulated. the background music key will change as the mouse is moved to drag the key control to a different setting. Similar control may be exercised over "echo". "tempo". "pan" and any other desired aspect of the background music. The "pan" control permits manipulation of the left/right stereo balance.

In addition. the music editor of the present invention permits the music to be changed in real time while the music is being played back. If a play symbol is clicked in Figure 14. the background music is played and the window shown in Figure 15A is generated. The window is used to change "music volume". "tempo". "echo" and "key" in real time with playing music. For example. if the "key" icon is clicked. a control bar. such as a key bar in Figure 15. is generated and real time control may be exercised.

The original model software includes a predetermined number of original sound waveforms associated with the game. e.g.. 32. The user is able to assign the original sound waveform to eight sound courses and selectively modify any aspect of the sound courses using the musical editor. A figure identifying each sound course on the sound sheet may be displayed by a unique mark associated with each original sound waveform. The musical editor software continually checks the state of each of the icons shown in Figure 15A and modifies the output music accordingly. In addition. the sampling waveform recorded from microphone 10 can be added to the original sound waveform list.

The game processor system additionally includes an "auto programmer" feature by which movement of an object may be modified by the user. An exemplary display explaining the operation of the auto programmer is shown in Figure 16. The auto programmer is accessed by clicking on the auto programmer icon 76 in Figure 7 or via the auto programmer icon associated with the "all status" editor of Figure 9. Initially. a unit such as the character shown in Figure 16 is selected by pressing the left mouse button. Using the mouse. a path containing up to 64 movements indicated by dots can be stored. If the left mouse button is depressed while the mouse is still moving along a path the entire path is stored. If the dots are disposed such that they are close together. the movement is slow. If the dots are further apart. the movement is faster. The speed of movement. however. can be further adjusted by the use of a tempo control shown in Figure 16. Figure 16 has an associated play or stop control in which a path can be played back or. if desired by the user. stopped and the path thereafter al-

A "unit collection" feature may be utilized to access a unit list identifying the various units that exist throughout the game. The unit list may be edited by the all-status editor. The list compiles all units which are registered within the game program. The tool box screen additionally permits access of a sound collection feature which provides access to a background music list which may be edited by the music editor and a special effects (SE) list which may be edited by the special effects editor.

The tool box select screen also permits access to a "game structure" collection feature. These features typically apply to the entire game and go beyond individual unit applicability. For example. a "game structure" feature may identify how many times a particular character may die before the game ends. This characteristic is set during game structure editing. In regard to the "game info" feature. game structure in this category may identify how many green turtles may appear in the game or what the first screen level may be. Such information may be edited via the data editor. The stage list game structure information relates to various game sequences which are available. Stage list information may be modified via the status editor. the program editor or the data editor.

Figure 21 identifies editorial/game fabricating changes that can be made during game play. After a system break is initiated. as described in detail above. a user may access editors by selecting. for editing. a moving object unit or by selecting a stage icon for changing the background. The various editors which are shown correspond to the icons associated with the various screen displays. For example. the animation editor. dot editor. auto programmer. special effects editor. all-status editor. key editor. status editor. program editor and data editor are accessible using the icons shown. for example. in Figure 7. Similarly. the stage or background related editors such as the music editor. status editor. screen editor. unit screen editor. map editor and data editor are associated with the stage window related icon shown in Figure 18. As shown in Figure 21 in the exemplary embodiment. the unit screen editor may be reached by accessing first either the screen editor or the map editor. The editors which are accessed via other editors are typically those which are utilized less frequently by. for example. more advanced game developers who may use a program editor to actually change game program language instructions.

The editors identified in Figures 20 and 21 operate in the manner described in conjunction with Figures 5 through 18. For example. the auto programmer editor shown in Figures 20 and 21 provide the ability to edit object movement as indicated on the display screen shown in Figure 16. The status editor identified in Figures 20 and 21 operates in the manner described in conjunction with the screen display of Figure 17. The animation editor identified in Figures 20 and 21 operates as described in conjunction with Figure 10. The all-status editor identified in Figure 21 operates as previously described in conjunction with the display screen of Figure 9.

Turning back to the hardware associated with the illustrative embodiment of the present invention, Figure 22 shows a more detailed block diagram of the gate array 226 shown in Figure 2A. The gate array 226 is responsible for controlling a variety of functions including superimposing the video signals output from CPU 228 and game CPU 200, game processor debugging related functions (hardware and software breaks. address. data traps. etc.) as well as a variety of interface functions (such as game CPU 200/main CPU 228 interface control: game hand controller interface and mouse. keyboard and ID card interface). The gate array 226 also is involved in CPU 228 bus access control. interrupt control. CPU 228 ready control: DRAM control including control of the control bus. address multiplexing. refresh control and peripheral LSI access control including control of the ROM 234. flash memory 236. floppy disk control 240. etc.

As shown in Figure 22. the gate array interacts with various components shown in Figures 2A and 2B including CPU 228. CPU 200. PPU 212. etc. Address decoder 260 is coupled to the address bus of CPU 228 and generates chip select signals to other devices associated with CPU 228. As can seen in Figure 22. address decoder 260 receives various clock (CLK). control signals (R/W). timing signals (BCYST) and generates the chip select signals for selecting the A/D converter 238. flash memory 236 and other components on the main CPU 228 as well as other gate array components.

Gate array 226 additionally includes memory controller 262 which is coupled to the CPU 228 address bus for generating address and other controls signals for controlling DRAM 230 and for generating various read/write signals associated with CPU 228. The memory controller 262 is responsive to clock. timing and control signals from CPU 228 to generate DRAM addresses MA01 through MA10. row and column address signals for the DRAM 230. a write signal for the DRAM 230 and I/O read and I/O write control signals.

Gate array 226 additionally includes a wait controller 254 which generates a delay for compensating for CPU 228"s very high access speed. The wait controller 254 is responsive to a timing signal (such as the bus cycle start signal (BCYST)) and clock and address signals from CPU 228. and. after the appropriate delay interval. couples a ready control signal to CPU 228.

Gate array 226 additionally includes an ID card interface 264. a printer interface 266. and a keyboard. pad. and mouse interface 268. The ID card includes an EPROM from which data is read. The ID card interface 264 interfaces communications between CPU 228 and ID card 6. The printer interface 266 is an interface provided for interfacing with a printer and includes input/output registers and other conventional printer interface circuitry. In the preferred exemplary embodiment. the printer interface 266 interfaces with a Centronics lems which may develop under rare circumstances due to removal or insertion of a cartridge from the game processor system console 2 while the power is on. As shown in Figure 24B. to preclude the unlikely possibility of cartridge components being damaged due to insertion or removal while the power is on. the pins in the cartridge edge connector are modified such that a pin which enables the cartridge RAM memory is shortened with respect to the power pin.

Figure 24B shows the ground. address. data. reset. and power connectors for the cartridge 4. The reset pin from the game cartridge is shorter than the powerJine pin so that when the cartridge is pulled out even before the signal lines are disconnected. the enabling "reset" signal prevents the cartridge RAM from being written to prior to the power being disconnected. When the cartridge is inserted, the power and data lines are connected prior to the reset signal being placed in a high state to thereby enable writing to RAM 336 as a final operation. In this fashion, the information stored in the cartridge working RAM 336 is protected. When the cartridge is inserted into its connector. power is applied prior to enabling of the cartridge memory circuitry. When the cartridge is removed, the cartridge circuitry is disabled before the power is disconnected due to the shortened enable signal receiving pin.

Turning to Figure 24A. when the cartridge is fully inserted with the power on. the reset signal (RES) is high. The reset signal is coupled to the enable input of a decoder 330 via a gating circuit 331. Decoder 330 includes one output coupled to a chip select pin of the game cartridge static RAM memory 336 (which is schematically shown in Figure 24A). A further decoder 330 output is coupled to the chip select pin of four bit binary counter 332. The binary counter is initially set at 0. Decoder 330. which is coupled to the game cartridge address bus. detects a request to write to counter 332 and initiate the counter 332 to count up. When the writing pulse is given fifteen times. the counter 332 state is 1111 and counter 332 generates an output signal which is coupled to gating circuit 334. If an attempt is made to write into SRAM 336 when the output of counter 332 is at a high level. a write signal is coupled to (each of the RAM modules which constitute) RAM 336 to thereby permit data to be written into the cartridge RAM. Thus. the cartridge includes a write protect circuit to require writing to a predetermined address area fifteen times before permitting writing data into the RAM to preclude any possibility of component damage or loss of data when the cartridge is inserted or removed with the power on. When an attempt is made to write to a predetermined address fifteen times. the cartridge memory system is in effect unlocked. In the game processing system of the illustrative embodiment. the time spent writing into the RAM cartridge 4 is very small.

Figure 24A also shows a cartridge edge connector 329. Among the signals coupled to the cartridge 4 are data signals and address signals carried on a data bus and address bus. Additionally. a reset signal is received. CPU read/write control signals and other control signal as received. The RAM cartridge shown in Figure 24A additionally includes a security chip 333 which operates with a security chip 227 in Figure 2A in a manner described in applicant's U.S. Patent No. 4.799.635 which patent is hereby incorporated by reference. As indicated above. further details of an illustrative security system employed in conjunction with the present invention are described in the concurrently filed application entitled "SECURITY SYSTEMS AND METHODS FOR A VIDEOGRAPHICS AND AUTHENTICATION GAME/PROGRAM FABRICATING DEVICE". Serial No. 08/332.812 filed October 31. 1994. After RAM cartridge 4 is fully programmed for use in an SNES game system. the security chip 333 may be used as part of a security system to determine whether the cartridge is an authentic game cartridge in conjunction with the SNES game console that includes a corresponding security chip 333. Additionally. CPU 228 receives the output from security chip 227 and prevents game program execution and editing by CPU 200 unless there is a determination of authenticity in accordance with the teachings of "635 patent. In this regard both the security chip 333 and the main CPU 228 perform processing operations as generally disclosed in U.S. Patent 4.799.635. Data indicative of the results of such processing are exchanged in a secure manner. If the main CPU 228 determines that the RAM cassette is not authentic based on data received via the gate array 226. the game CPU 200 is maintained in a reset state.

Figure 24C is another illustrative way to implement cartridge circuitry corresponding to Figure 24A. Circuits in Figure 24C which correspond to like circuits in Figure 24A are identically labeled and will not be further described. The one shot multivibrator generates the load signal to counter 332 when the CLK21M input carries no signal. The counter 332 is set to 0 by the load signal and writing into SRAM is disabled. In the normal operation. the CLK21M signal is continuously supplied from the console so the load signal is always disabled and the SRAM access can be accomplished in the same way as the circuitry of Figure 24C.

Figure 24D shows an illustrative embodiment for cartridge connectors for use in conjunction with Figure 24C. When the cartridge is pulled out. the CLK21M and reset signals are cut off before the other indicated signals are cut off. If the CLK21M input carries no signal. the load pulse is enabled. Thereafter. counter 332 loads 0 data and writing into the SRAM is disabled. In Figure 24D. the address bus and data bus connectors are shorter than GND and VCC pins. Therefore. "LATCH UP" is prevented when the cartridge is pushed in while keeping power on.

Figure 25 is a block diagram of reset related circuitry in accordance with an exemplary embodiment of the present invention. The system includes a reset button 1 (347) and a reset button 2 (348). i.e.. one reset button drive. If the floppy disk diskette is not in floppy disk drive 199. then a message is displayed to the user indicating "please insert the tool disk" (389). Upon detection of the tool disk being inserted. a tool installation screen is generated (390) and the Mario Factory loading is completed.

In accordance with one exemplary embodiment of the present invention. the model software is loaded such that two distinct files are stored: a base file containing non-modifiable aspects of the game and a user file containing those elements of the game that a user may modify. Alternatively. only the base file need be initially loaded to provide the user with the fundamental aspects of game play from which to build upon and create new games within the genre of games associated with the base file.

The check at block 392 determines whether the model software is in the floppy disk drive. If the model software is not in the floppy disk drive 199. the user will be prompted to install the model software disk (394) and a model software load screen is generated (393). Thereafter a check is made if a user file has been selected by the user (395). The user file may be loaded on a different disk than the base file (which is why the selection of the user file in block 395 may require the insertion of another disk). The user file contains the changes made to the model software video game to date. Initially the user file is in a default stage where the model software defines the initial game play. As the user changes the game. the default information is changed to reflect the user's editorial modifications. Both the game cartridge and the floppy disk include a base file and user file. however. the floppy disk will store the original default version of the game whereas the game cassette will not. The RAM cassette data is modified in real time such that default data is not resident on the cassette 4. Once the user file has been selected. a user file load screen is displayed (398). If a user file is not selected. the user will be prompted to insert a user disk (396) and the user will have an another opportunity to select a user file (397). Thereafter a utility title screen (399) is displayed such as shown in Figure 5.

In block 392 the model software may be changed as desired by the user by inserting another disk. Once the utility title screen is displayed. a user may click a particular file icon which leads to branching back to block 395 in which the user can select another user file. Alternatively. the routine may branch back to block 392 in which the entire genre of games may changed by changing the model software. The Mario Factory program includes a large variety of editing tools such as "dot" for drawing particular characters "animation data" for controlling animation. etc.

The game processor system. in accordance with the exemplary embodiment described herein. includes software under the control of CPU 200 and software under the control of main CPU 228. The software operated by the CPU 200 includes the model game software described above and user fabricated game programs. The main CPU 228 executes various utility programs. operating system. peripheral driver programs. and BIOS and IPL software. The utility software operated by the main CPU 228 includes game editing tools. network software. word processing software, disk management software. etc. The operating system and peripheral driver software includes subroutines for supporting peripheral devices not fully supported by the BIOS software described below.

The BIOS software includes low level routines for direct interaction with the above described system hardware. More specifically. the exchange of information between the main CPU 228 portion of the system and the game CPU 200 subsystem is controlled from a software perspective by way of two BIOS operating system routines, one of which is resident in BIOS ROM 234 and the other which is resident in monitor ROM 204. The operations performed during such information transfer by these operating system routines are described below. The BIOS routines control the various I/O devices shown in Figure 2A and 2B such as mouse 16. keyboard 18. controllers 12 and 14. display 15. modem 22. and control processor intercommunication. memory maintenance functions and PPU control functions. PPU 212 functions controlled by the CPU 200 BIOS software includes. for example. setting of the PPU's color generating RAM to define predetermined desired color maps. transferring data to the VRAM 214. setting superposition priorities. etc. Screen display related functions that are controlled include setting the cursor position. Keyboard functions that are controlled include keyboard initialization. obtaining keyboard buffer data and obtaining keyboard status information. Similarly. for the mouse. a variety of functions controlled by BIOS software including mouse initialization. obtaining the cursor position of the mouse. etc. BIOS software associated with controlling operations of the game CPU 200 include initialization of the game CPU. transferring data to the sound processor 208. working RAM 210. and functions associated with the exchange of data with the main CPU 228. The BIOS software associated with the main CPU 228 includes controls functions such as setting the real time clock. checking the time-out counter in gate array 226. and performing operating system housekeeping functions such as those which are typically performed in conventional operating systems such as MS DOS. The operating system software is capable of receiving some commands from other computers so that game programs received via the SCSI board 24 in Figure 2B from an IBM compatible PC may be executed and edited.

The IPL software is responsible for initial program loading such that when the system is booted up it checks the game processor system status and boots up the BIOS in the operating system routines. The initial program loading (IPL) routine is executed upon system start-up and initially results in the display of the start-up message. including a copyright notice as referenced controlled by a game controller 12. 14). a unit 2. which is labelled an "enemy" unit and a unit 3 which is a music unit. Significantly. all data and algorithms relating to a unit. such as graphic data. the appropriate responses to keys. and sound effects are stored for the unit. as is described below. This system also provides for music units which are not associated with graphic data. The game is edited by selecting a tool icon to make changes. For example. for Figure 28. if the unit 1 is selected. a tool selection may be made in which the Mario graphics sound or programmed responses to conditions. From a programmer's point of view. a unit may be considered as a memory space for storing game play characteristics and a wide range of other data allocated on the game CPU 200 side.

As previously described. the preferred embodiment of the present invention includes a wide variety of editing tools for changing pictures. sounds and programs. Editors that are specialized for certain games may be included in the model software as expansion software. For example. for a shooting game specialized map editors or enemy path editors may be utilized. Similarly. for a role playing game. specialized map editors. dialogue editors and event editors may be included and in a puzzle game. a rule editor may be included as expansion software.

Each of these various types of tools are displayed for selection by the user. Icons are organized in layers and only the icon that is needed at a particular moment is displayed to the user. Unnecessary icons are not displayed. rather. they are recorded in the status file included in the model software.

Game editing revolves around the editing units and associated data structures. In an embodiment of the present invention. different editing functions are controlled based upon icon clicking at system break or from the tool box screen. In another illustrative embodiment. they are controlled based upon whether the left or right mouse button is depressed. After game play has begun. a system break is initiated and a mouse cursor appears on the screen. If it is desired to edit a particular unit displayed on the screen. for example. the left mouse button is clicked over one of the units shown. for example. in Figure 28 and all the tools for editing the selected unit are displayed including graphics editing. program editing and sound editing features. If the right mouse button is clicked. various options are displayed such as "undo". "game". "file". "tool list". "unit" and "end". If the tool list option is selected. tools may. for example. be shown as a tree chart and the units that can be edited with the selected tools may be controlled to blink.

During this process. the edit screen is superimposed upon the game screen. After the desired edits have been made. game play may be resumed. In left mouse button edit operations. after a game screen is displayed and a system break is initiated. the mouse cursor appears on the screen and the user clicks on a character to be edited. The character to be edited appears in a box with the unit identified. Thereafter. modification of the unit takes place via the various editing tools. After editing. game play may be resumed.

With respect to right mouse button edit operations. on a combined screen a tool list is generated. The user then selects a tool and each of the editable units will be displayed for that tool. Thereafter. via the mouse. another tool may be selected and different editable "units" will be displayed. If a character is clicked on, the character appears in a box and a menu appears and editing continues in an identical manner as with left mouse button editing. In right mouse button editing. during Mario Factory operation, the status file is checked and the status is displayed. The unit header (which is a data area placed at the head of the unit storing information such as the tools with which the unit can be edited) is then checked and editable units are found.

In the present system. due to the model software selection screen, even without the model game software itself. the Mario Factory authoring tool can be started up and basic tools can be used. The Mario Factory program includes a status file, which is a text file which records the label names of tools to be placed in memory. By accessing the status file. the Mario Factory program can determine which tools need to be stored in DRAM 230. Some of the tools stored in DRAM 230 are displayed as icons Such tools may include basic tools and expansion tools chosen from an object file written in main CPU 228 code.

Figure 29 is a block diagram which functionally demonstrates how programming is structured on a "unit" basis. Within the memory address space 401 of game CPU 200 is stored unit related data uniquely identifying each of the units which may be edited. There is a one-to-one correspondence between units which may be edited and unit related data structures stored in the game CPU memory address space.

As shown in Figure 29. if a user selects unit 3 shown on display screen 15 during. for example, a system break operation. a corresponding portion of memory space 401 will include a unit data structure (or pointer) associated with unit 3 which will store (or point to) a unit header. a data table and a unit 3 related program. The unit header includes labels for tools that can be used to edit the particular unit and the storage address for edited data. The group of unit headers are also stored in the main CPU 228 address space. The data table includes data relating to a wide range of game play/object characteristic information to be described below including not only graphics and sound but also information indicative of the path movement for each character. In essence. the graphics. sound and program associated with each unit may be viewed as being carried with or pasted on each unit.

The main game processing routine includes a sequence of instructions for processing each of the units 1 to N. The main routine (405) includes a sequence of jump to subroutine (JSR) for accessing the respective information area and a game background related RAM area. As shown in Figure 33. the object unit information includes the object unit ID. status register and coordinate location and a wide range of other object information such as character size. pose information. etc. This information is stored in the buffer area of main CPU 228 and DRAM 230. which is shown in Figure 2A and is used to edit with the control file. The control file has some cross reference data. label text and message text for assisting in editing. The data stored in the control file is not needed for game program execution but. is used to make editing easier. The background music data that is utilized during game play is transferred from main CPU 228 system portion back to game CPU 200 section.

Figure 34 depicts main CPU 228 and game CPU 200 memory maps and identifies the manner in which data is extracted and copied from game CPU 200 to main CPU 228 during a system break. It also depicts how certain common values are maintained in each memory area during editing so that main CPU 228 and game CPU 200 can efficiently coact.

In the data area in the CPU 200 portion of Figure 34. graphic data. sound data and status (process program) are stored as animation data. ASM data and BASIC data. The memory area of game CPU 200 also includes the main program described in conjunction with Figure 29 and the associated the main processor communication routine in SHVC-BIOS. The uneditable base file that is provided with copy protect mechanisms is stored as a portion of the data area. unit pointer and the assembler main program. In the main CPU 228 memory system. in addition to the basic editing tools of the Mario Factory. the CPU code is stored. The memory area of CPU 228 also includes the main program area to store the operating system program and main CPU communication routine. Additionally. the base file is stored as the control file and some portions in buffer areas including unit data. unit pointer and certain minor data for editing. These elements are loaded via the load program controlled by the user's model software selection in the Mario Factory program.

When the Mario Factory is executed. the execution results in the allocation in the main CPU 228 memory as shown in Figure 34 and the display of the title screen. Then. if the load icon is clicked. certain files of the model software are loaded in CPU 228 memory and certain files are loaded to the game CPU memory via the communication routines in each memory. The model software includes some unit information written to the data area and unit pointer in SCPU 200 memory. The model software also includes the control file data that is the minor data for editing as cross reference data. label text or message text which is stored in the control file area in CPU 228. The control file data includes the attribute data to indicate whether the edited data is base file data (disable the edit) or user file data (enable the edit). In addition. the control file data includes the limit data to indicate the limit of editing. For example. a limit may be set specifying that Mario speed cannot be above one frame/8 dots. In accordance with an illustrative embodiment of the present invention. the user file includes the data area data and unit pointer data except for unit information of the base file. The user data also may include the differences between the default data and edited data.

The flowchart of Figure 35 delineates the sequence of operations performed by the game CPU 200 and the main CPU 228 during game fabrication processing. It illustrates how a game program is executed. stopped for making game modifications and then resumed from the point at which editing occurred.

The left portion of the flowchart depicts the flow for game CPU 200. whereas the right side of the flowchart indicates the flow for main CPU 228.

The flow of main CPU 228 is shown in parallel with the game CPU flow to demonstrate the interaction and data exchange. Such interchange results from the stopping of a game screen and the selection of a particular unit for editing. e..g. Mario. The arrows between the columns indicate an exchange of data between the two processors.

Initially. once game play is to commence. the game CPU 200 is in control and begins executing the main routine for playing a game embodied in the model software as previously described (433). When a system break is initiated. a COP command is generated. as previously described. to cause a vector jump resulting in the software being executed out of the monitor ROM 204. When a system break is imposed and. for example. the character Mario is selected for modification. a unit identifier is generated. Information which is being changed during the system break is stored in the CPU work RAM 202. After the vector jump. game CPU executes out of the monitor ROM 204.

As shown in Figure 35. the main CPU 228 executes a program loop to be described below. awaiting a stop request from the user for making a change in the model software game play. When the mouse button is depressed by the user. a stop request is detected (440) and a software interrupt occurs in which a COP command is coupled to the game CPU (441).

If the system break is imposed while the screen display is being created. the generated display will be compromised. The SCPU BREAK LOGIC shown in Figure 26 delays the system break until no impact is possible on the generated displays such as at the end of a nonmaskable interrupt (NMI) which occurs. for example. during the vertical blanking interval. When main CPU 228 provides a COP command to CPU 200 via the SCPU BREAK LOGIC at the end of the NMI interval. CPU 200 begins executing out of monitor ROM 204 to transfer data to be changed to main CPU 228 buffer area. The main CPU 228 is informed of the coordinate position of the character to be changed. the size of the object and the identifying unit number. The unit work RAM area of CPU 228 includes status information relating to cordance with the must subroutine shown in Figure 40 (475): a flag called "BINGO" is set to 1 (477) which means that the condition always will occur and the routine returns to the main table processing routine (479).

Figure 41 shows the "Enemies Wipe-Out" condition subroutine (480) shown in the main table. This routine determines whether the enemies are eliminated and sets the "BINGO" flag. Figure 41 also represents the "All Friends Wipe-Out" condition subroutine to determine whether all friends are destroyed (480). In accordance with the enemy (or friend) wipe-out routine. a check is initially made to determine whether the number of enemies (or friends) is equal to 0 (482). If the number of enemies (or friends) equals 0. then the BINGO flag is set to 1 (484) and the routine branches back to the main table processing subroutine. The Go Next Stage subroutine in Figure 44 (or Go Ending subroutine in Figure 45) is then executed in process 466 in Figure 38B. If the number of enemies (or friends) are not equal to 0 then the BINGO flag is set to 0 (486). Thereafter. the routine returns to the main table processing routine. The corresponding process is not executed.

In accordance with the "unit action" subroutine process in the main table (500). a unit action table is processed such as the illustrative unit action table shown in Figure 42 which. for example. describes actions relating to a "turtle" unit work. Focusing on the conditions specified in the unit action table. each of the conditions as well as the processes are identified by address pointers to a subroutine responsible for the condition and processing. The "must" condition routine has previously been explained in conjunction with Figure 40. The condition following "must" relates to when the turtle is detected touching the right edge of the screen. The "BE-TREADED" condition refers to when the turtle is stepped on by another object. When the condition on the left side is met. the process indicated by the associated address pointer is executed. The processes associated with each of the conditions in the table shown in Figure 42 are self-explanatory.

In accordance with the "unit action" subroutine shown in Figure 43 (500). initially the routine references the unit data in the unit work area in Figure 34 (502). Thereafter. the contents of the unit action table shown in Figure 42 are processed by accessing the subroutine indicated by the routine address associated with each of the conditions (504). The unit action table may be modified by the user through the status editor using the display screen shown in Figure 17. A check is thereafter made to determine whether a condition has been met (506). If the check at block 506 indicates that the condition has been met. then the process portion of the table in Figure 42 is accessed to perform the associated processing such as. for example. move the turtle to the right (508). If the first condition associated with the unit action table is not met then the routine branches to block 510 where a check is made to determine whether the table is at an end by detecting the end mark. If the table has not ended. then the routine branches back to block 504 and processing continues. If the table is at an end. then a check is made at 512 to determine whether there are no more units to be processed. If this check does not reveal that there are no more units to be processed. the routine branches back to block 502. If there are no more units to be processed. then the routine returns back to the calling routine (514).

Turning back to the main table (in Figure 39). if the condition Enemies Wipe-Out is detected. then the "Go Next Stage" routine is executed which is shown in Figure 44 (516). Initially. the stage number is incremented to initialize for the next stage (518). Thereafter. the unit work area in Figure 34 is cleared (520). some units are set and the background picture is displayed for the next stage (522). The next main routine is set to the next stage routine. Thereafter. the routine branches back to the calling routine (526). The last process shown in the main table is the "Go Ending" process which is executed upon detection of a friend"s wipe-out condition. As shown in Figure 45. when the Go Ending subroutine has been called (528) the unit work is cleared (530). Thereafter. units are set for ending (532) and the associated picture is displayed (534). The routine then branches back to the calling routine (536). The next main routine is set to the ending stage routine.

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment. it is to be understood that the invention is not limited to the disclosed embodiment. but on the contrary. is intended to cover various modifications and arrangement included within the spirit and scope of the appended claims.

Claims

1. An interactive computing system for editing a videographics program comprising:

   a first processor that is operable to execute a videographics program and to generate a video graphics display output:
   a second processor that is operable to perform videographics program editing operations and to generate an editing related display output: and
   superimpose control circuitry for superimposing said editing related display output onto said videographics program display output.

2. A videographics program editing system according to claim 1. further including at least one picture processing unit coupled to said first processor and said second processor for generating a videographics display on said display screen.

3. A videographics program editing system according

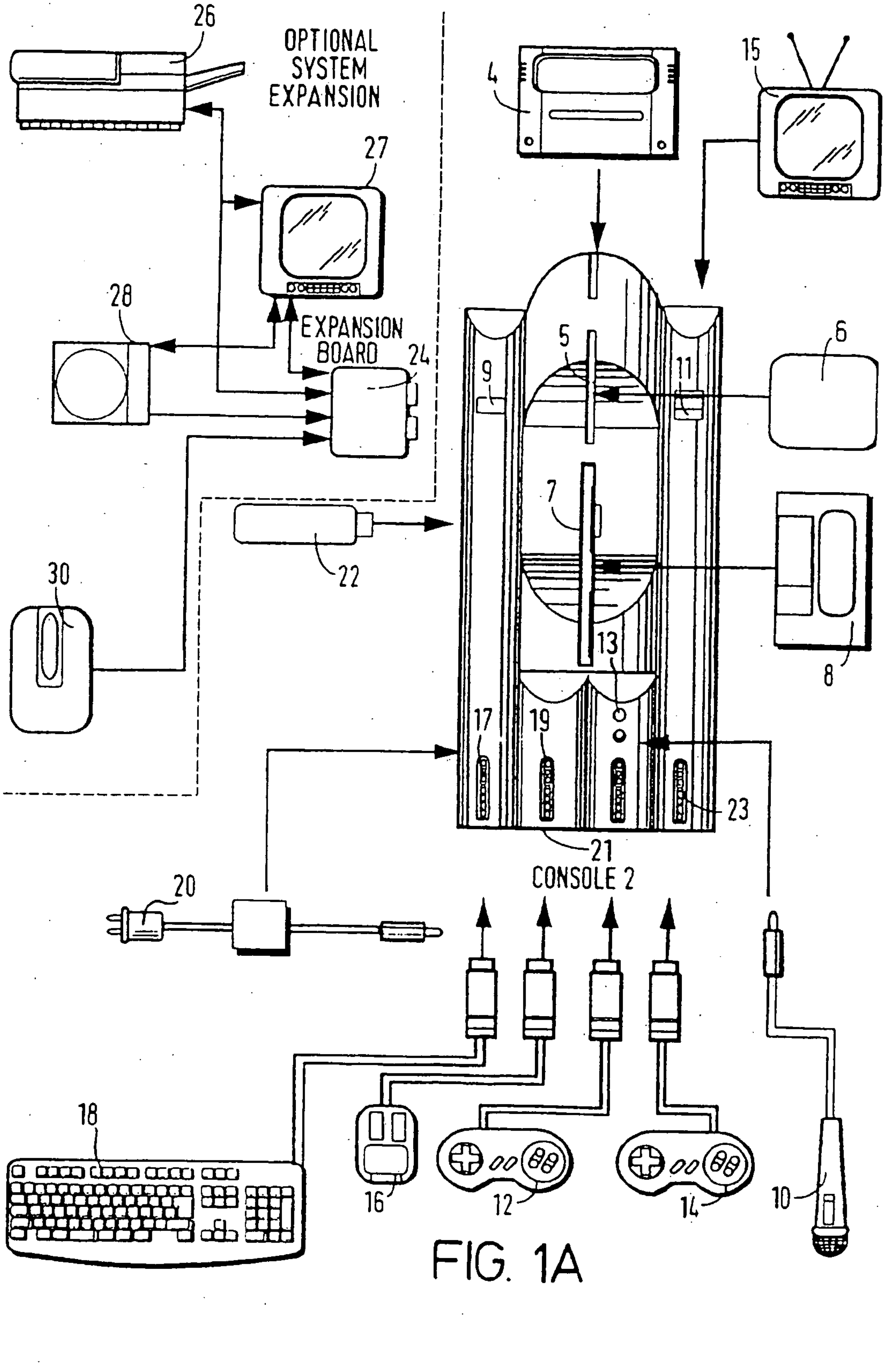
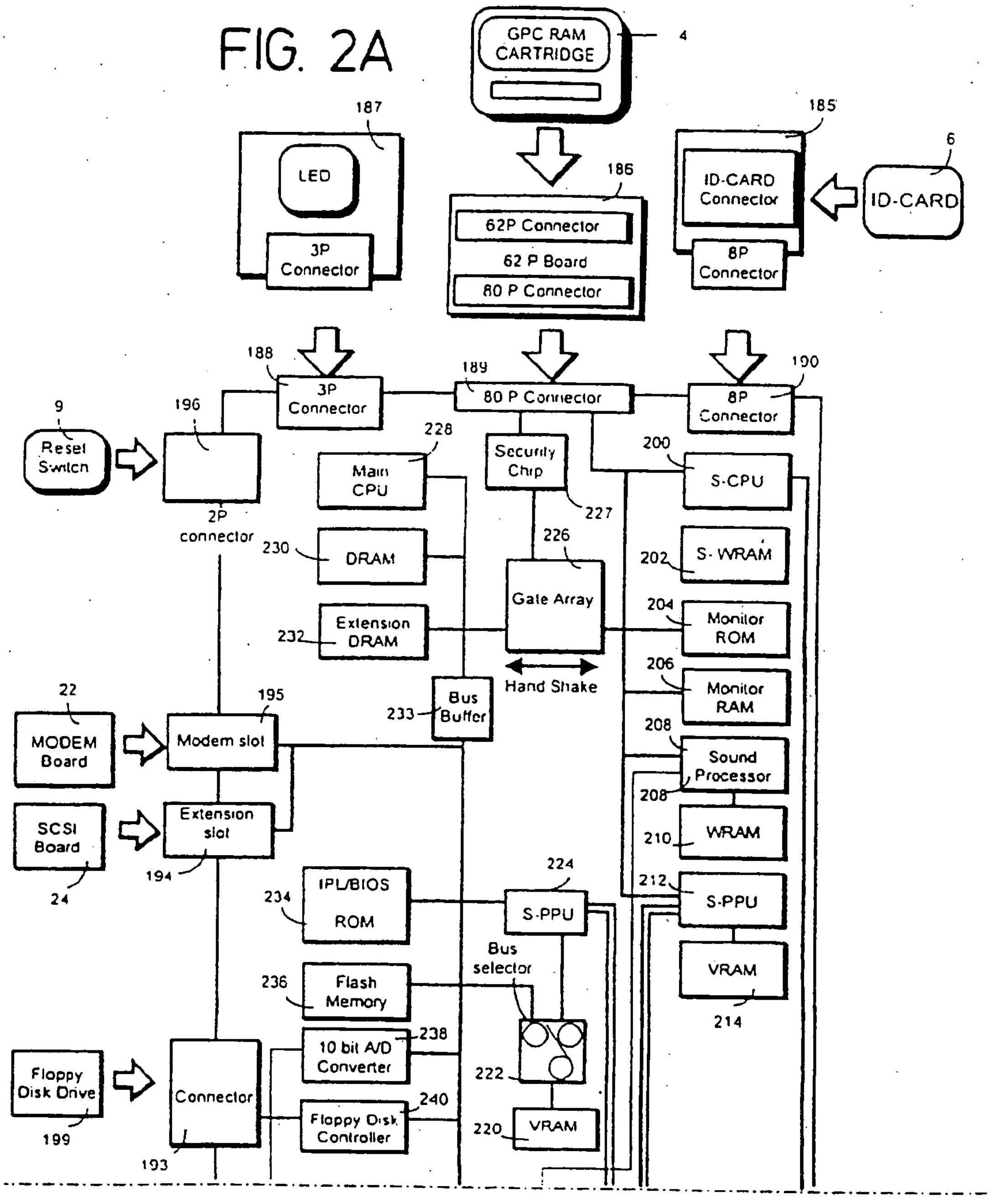

26
OPTIONAL SYSTEM EXPANSION
27
28
EXPANSION BOARD
24
30
22
4
15
5
9
11
6
7
8
13
17
19
23
21
CONSOLE 2
20
18
16
12
14
10

FIG. 1A

FIG. 2A



TO FIG. 2B

FIG. 3A

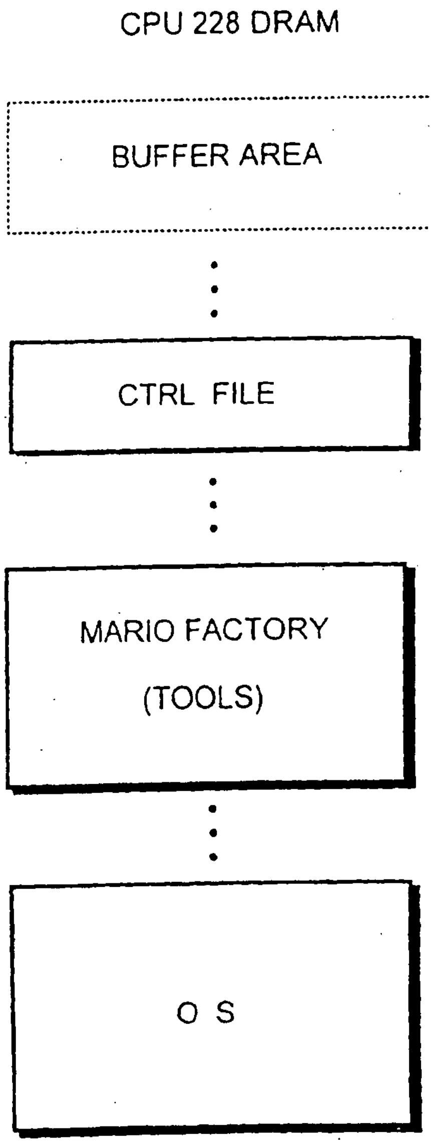

CPU 228 DRAM
BUFFER AREA
CTRL FILE
MARIO FACTORY
(TOOLS)
O S

FIG. 3B

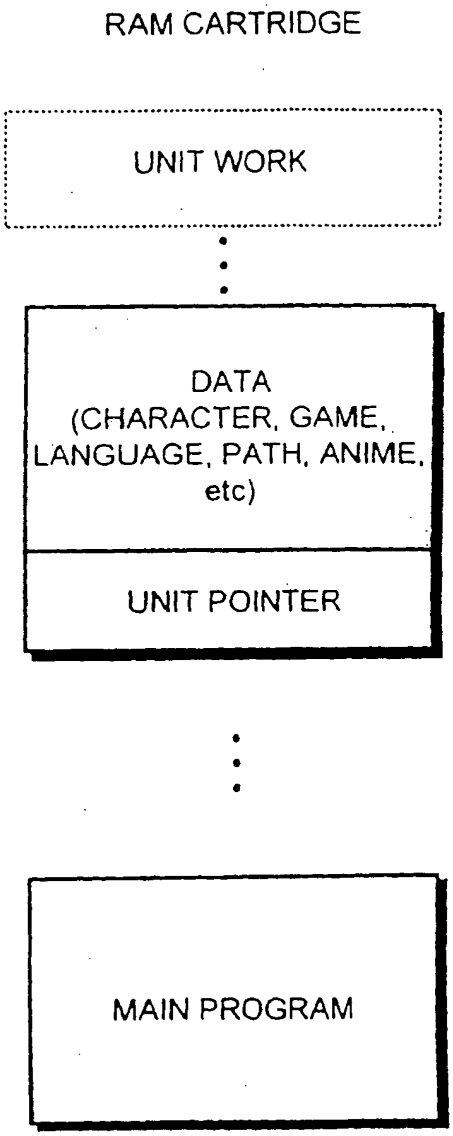
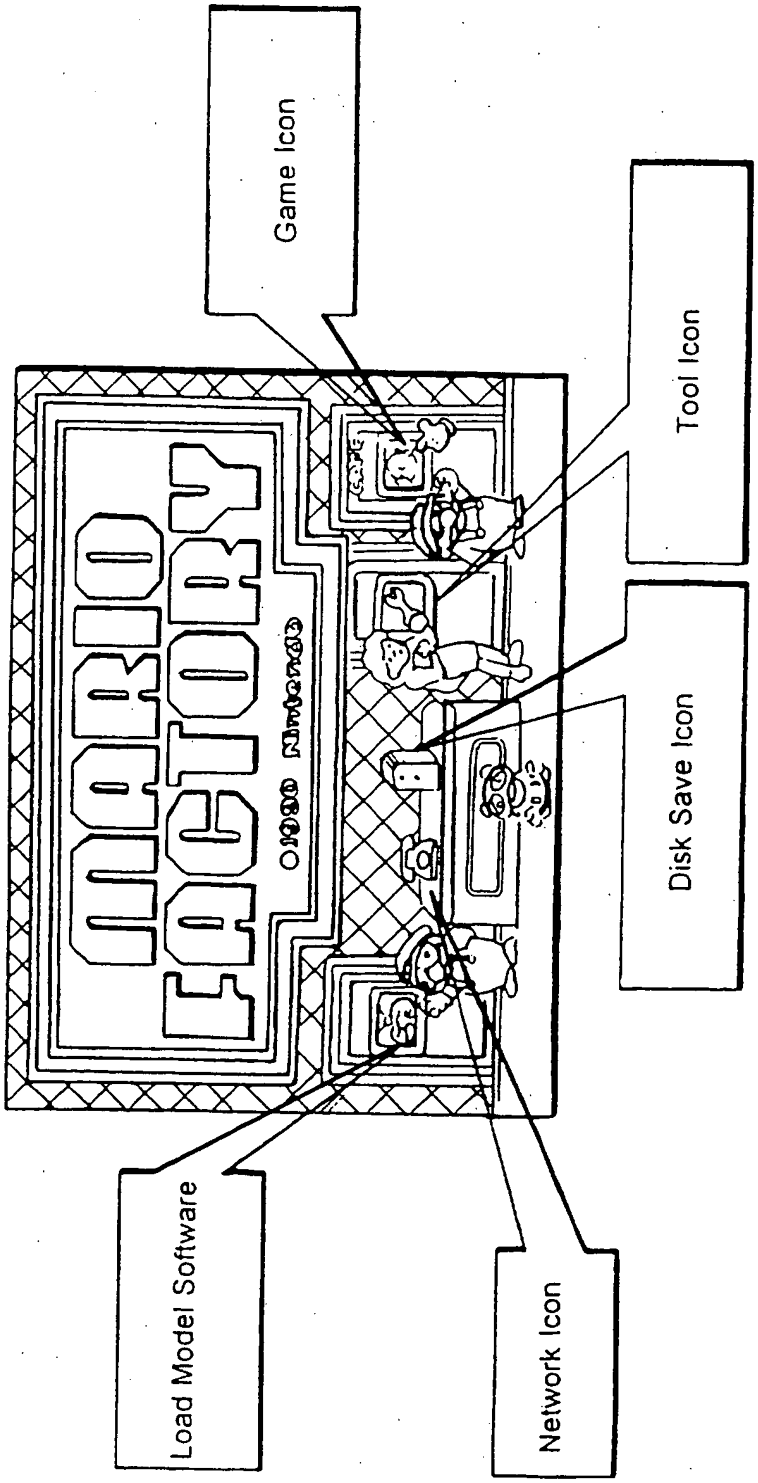

RAM CARTRIDGE
UNIT WORK
DATA
(CHARACTER, GAME,
LANGUAGE, PATH, ANIME,
etc)
UNIT POINTER
MAIN PROGRAM

FIG. 5


Game Icon
Tool Icon
Disk Save Icon
Network Icon
Load Model Software
MARIO FACTORY
©1990 Nintendo

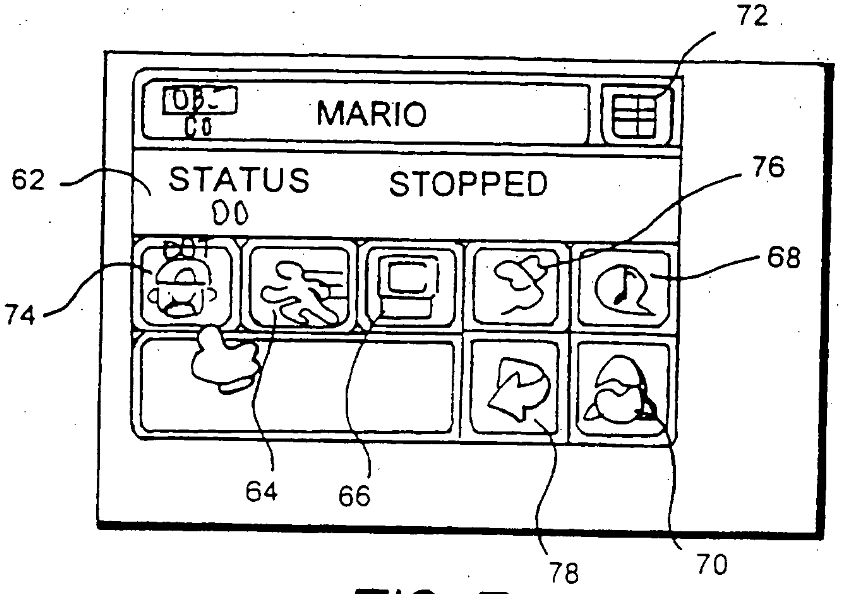

72
OBJ
CO
MARIO
62
STATUS
DO
STOPPED
76
68
74
64
66
78
70

FIG. 7

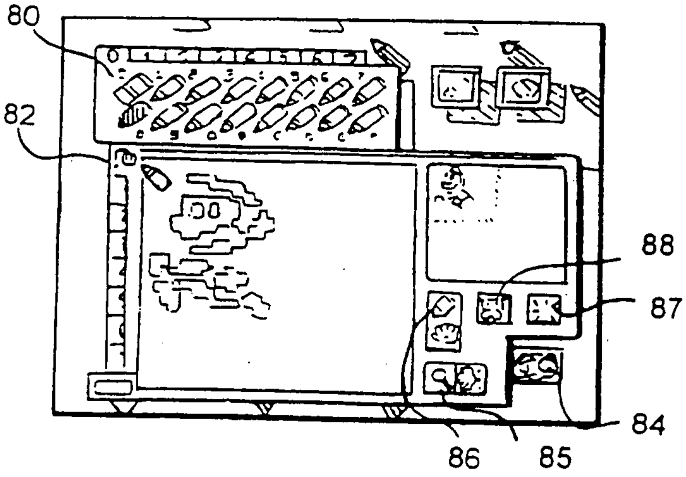

80
82
88
87
84
86
85

FIG. 8

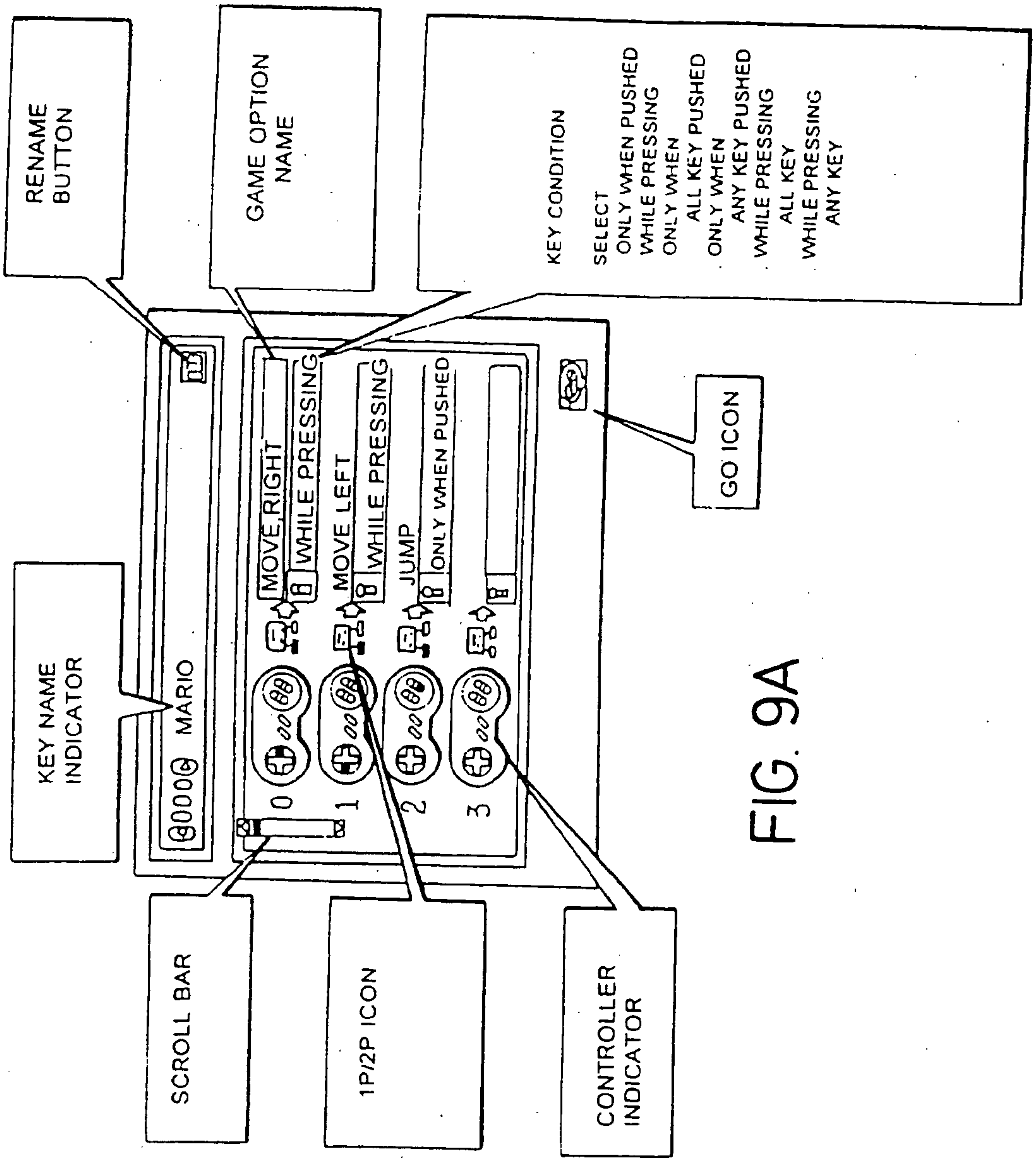
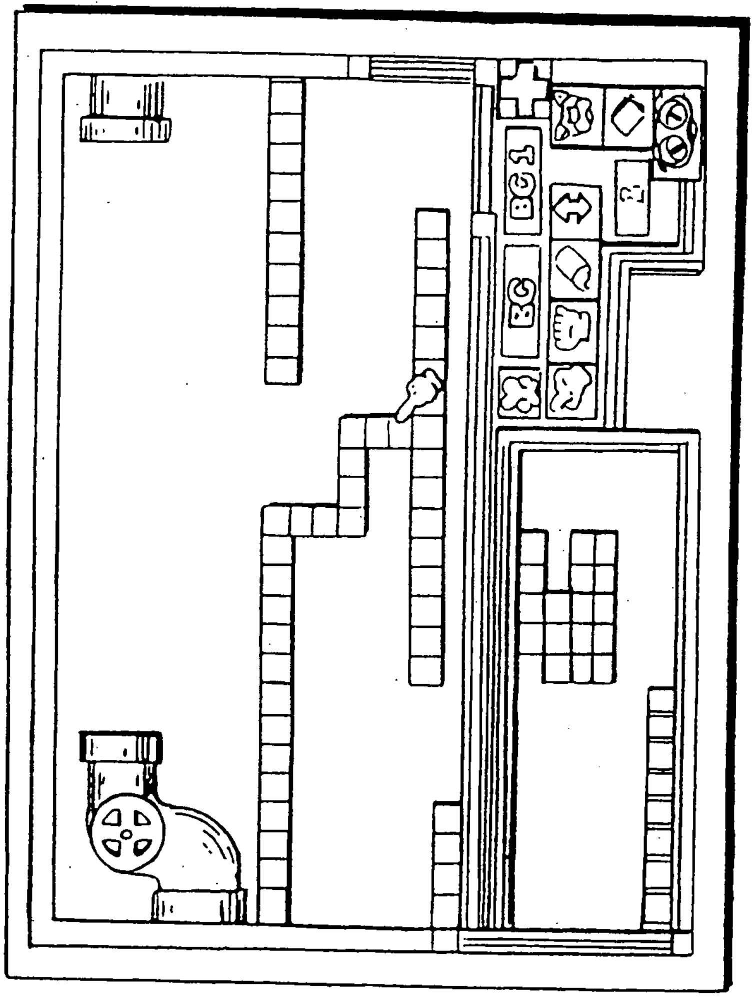

RENAME BUTTON
GAME OPTION NAME
KEY CONDITION
SELECT
ONLY WHEN PUSHED
WHILE PRESSING
ONLY WHEN
ALL KEY PUSHED
ONLY WHEN
ANY KEY PUSHED
WHILE PRESSING
ALL KEY
WHILE PRESSING
ANY KEY
KEY NAME INDICATOR
0000G MARIO
MOVE RIGHT
WHILE PRESSING
MOVE LEFT
WHILE PRESSING
JUMP
ONLY WHEN PUSHED
0
1
2
3
GO ICON
SCROLL BAR
1P/2P ICON
CONTROLLER INDICATOR

FIG. 9A

FIG. 11



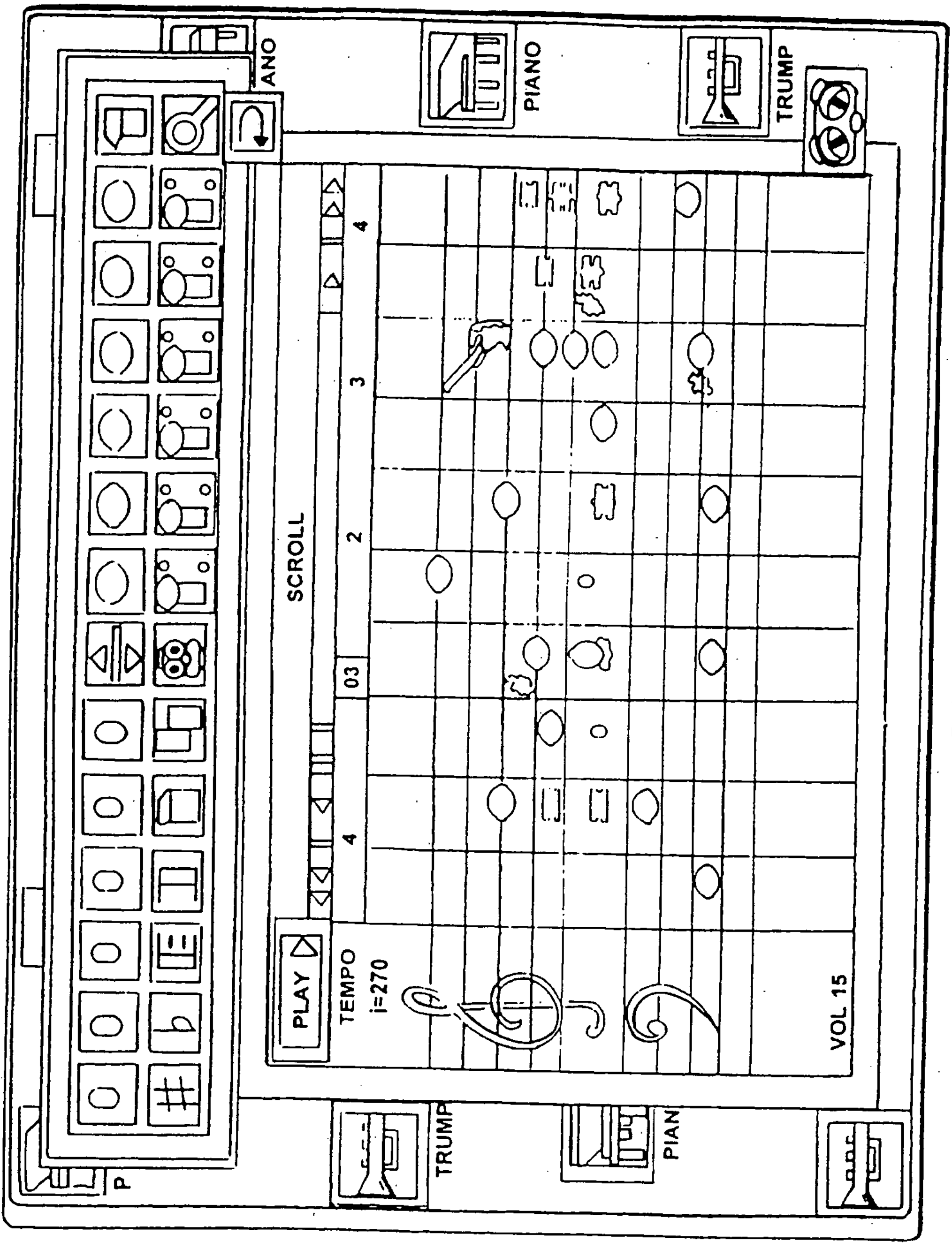

ANO
PIANO
TRUMP
SCROLL
4
3
2
03
4
PLAY
TEMPO
i=270
VOL 15
TRUMP
PIAN
P

FIG. 14

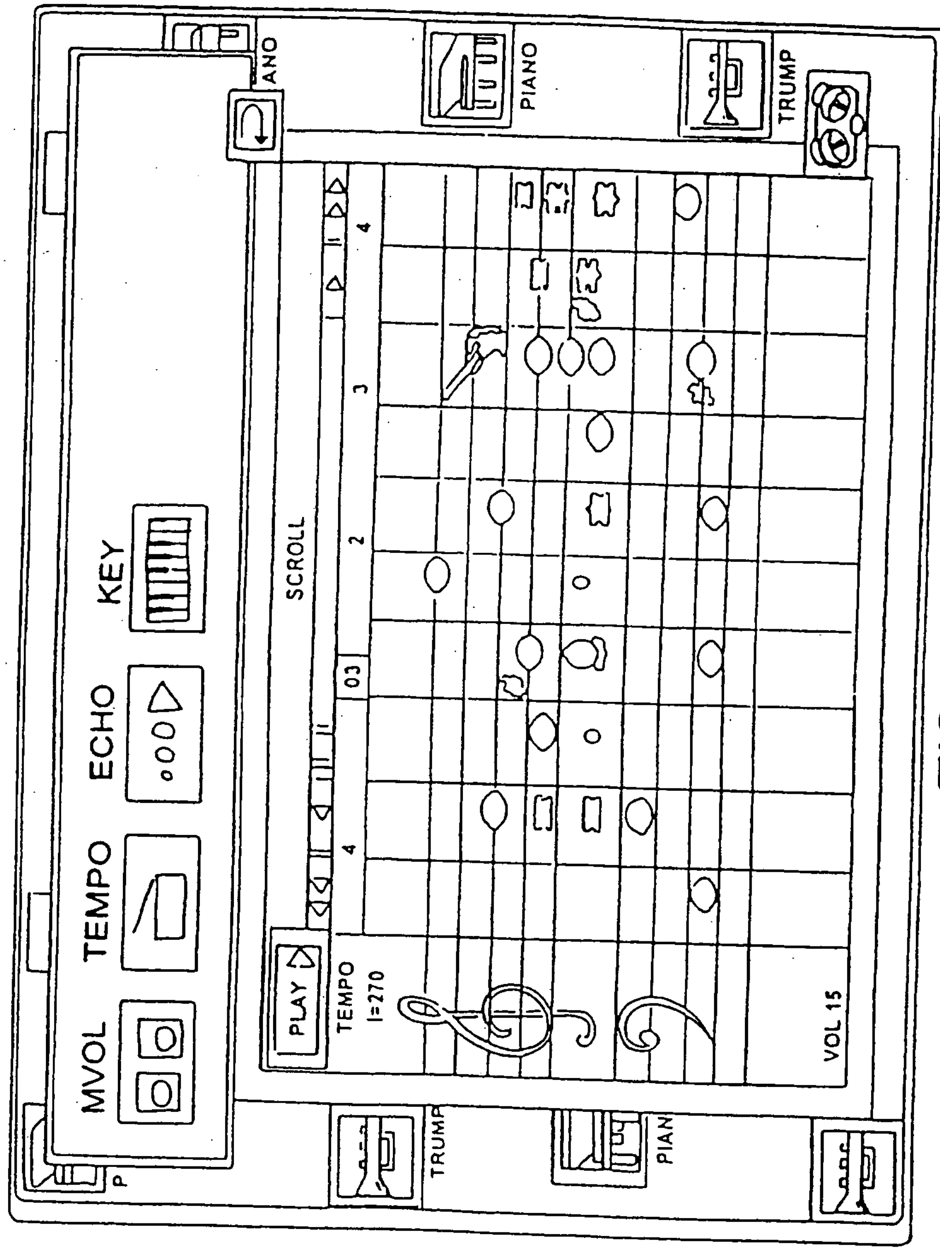

MVOL
TEMPO
ECHO
KEY
SCROLL
PLAY
TEMPO
I=270
VOL 15
4
03
2
3
4
ANO
PIANO
TRUMP
P
TRUMP
PIAN

FIG. 15A

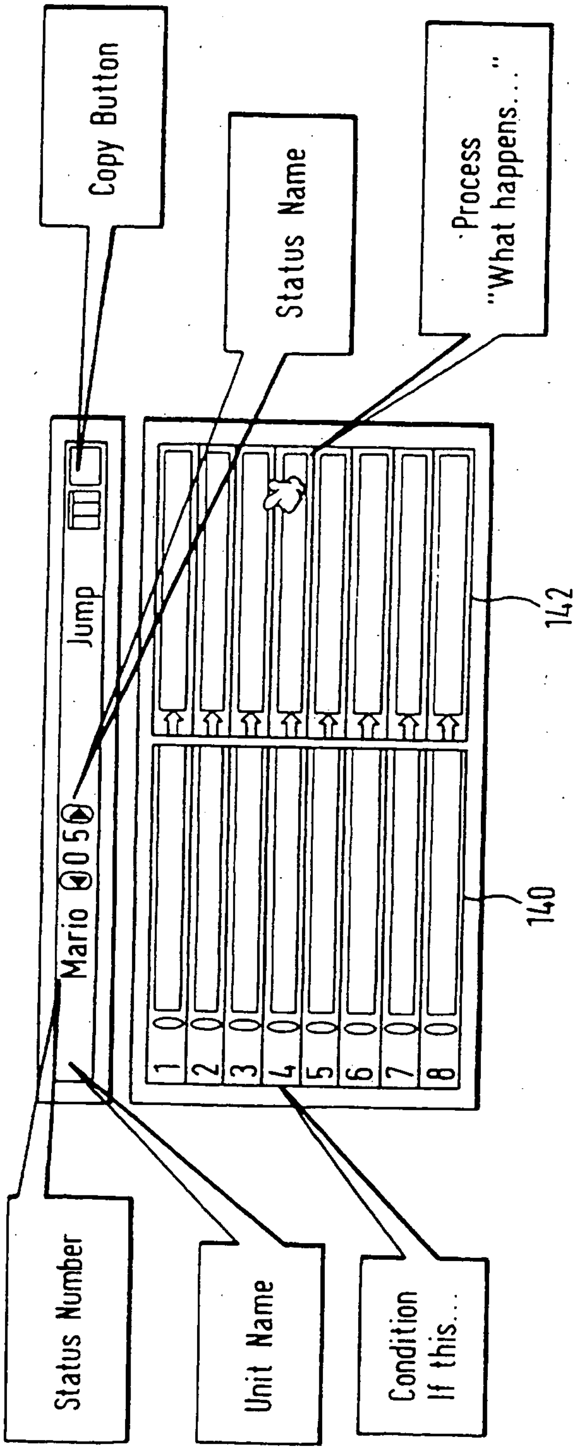

Copy Button
Status Name
Process
"What happens... "
Jump
Mario
05
142
140
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
Status Number
Unit Name
Condition
If this...

FIG. 17

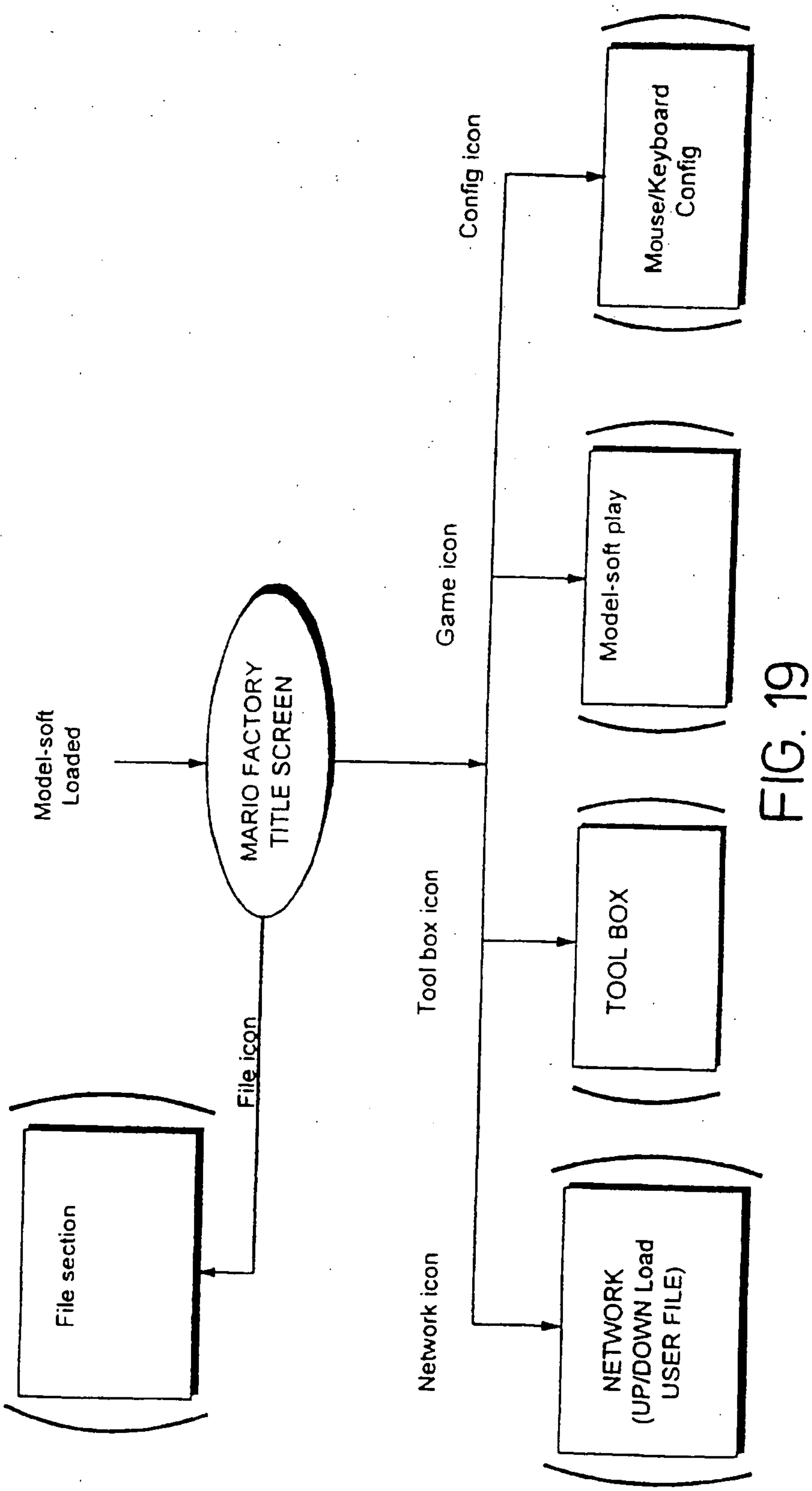

Model-soft
Loaded
MARIO FACTORY
TITLE SCREEN
File icon
File section
Config icon
Mouse/Keyboard
Config
Game icon
Model-soft play
Tool box icon
TOOL BOX
Network icon
NETWORK
(UP/DOWN Load
USER FILE)

FIG. 19

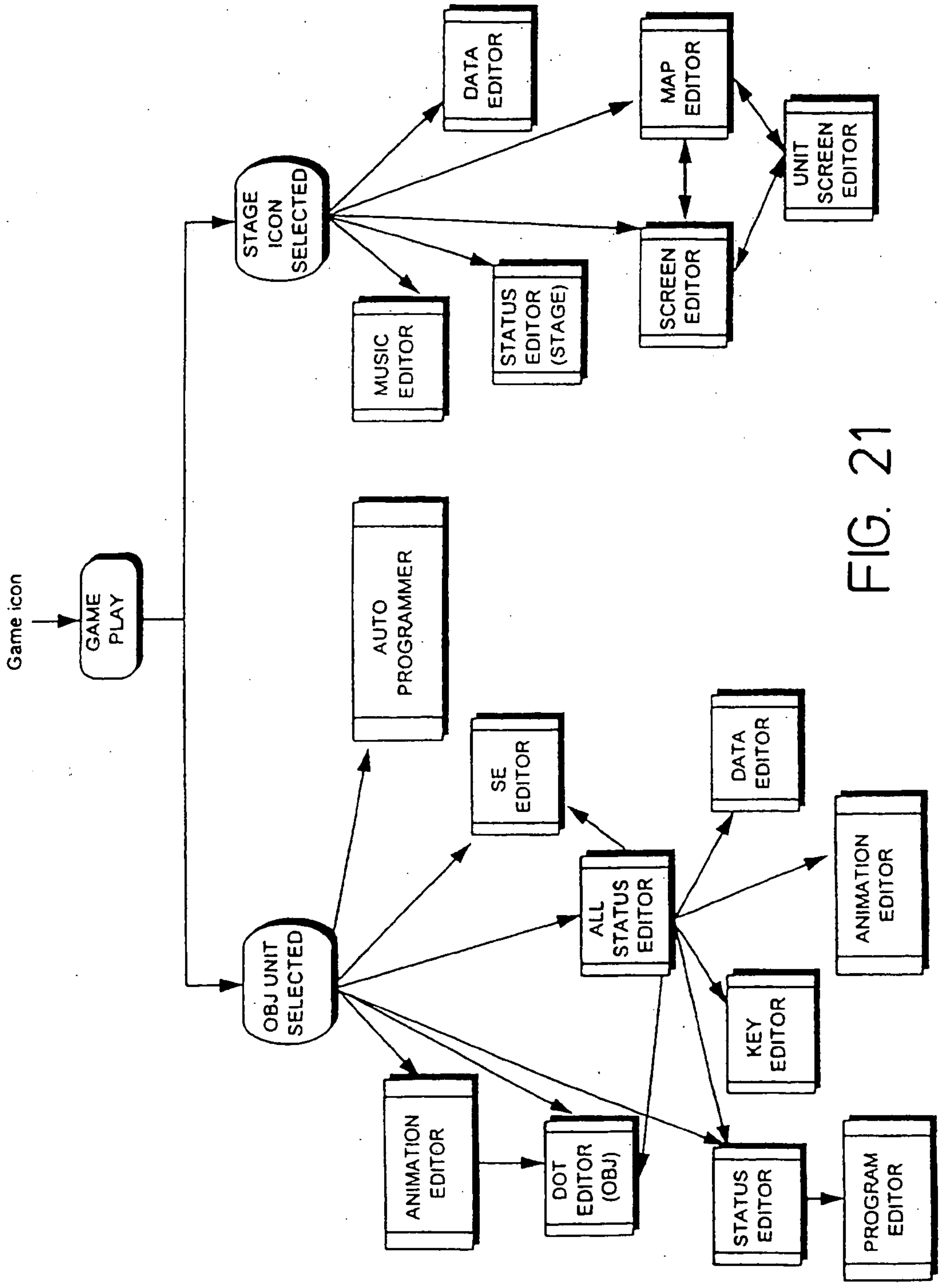

Game icon
GAME PLAY
STAGE ICON SELECTED
DATA EDITOR
MAP EDITOR
UNIT SCREEN EDITOR
SCREEN EDITOR
STATUS EDITOR (STAGE)
MUSIC EDITOR
AUTO PROGRAMMER
OBJ UNIT SELECTED
SE EDITOR
ALL STATUS EDITOR
DATA EDITOR
ANIMATION EDITOR
KEY EDITOR
ANIMATION EDITOR
DOT EDITOR (OBJ)
STATUS EDITOR
PROGRAM EDITOR

FIG. 21

FIG. 23

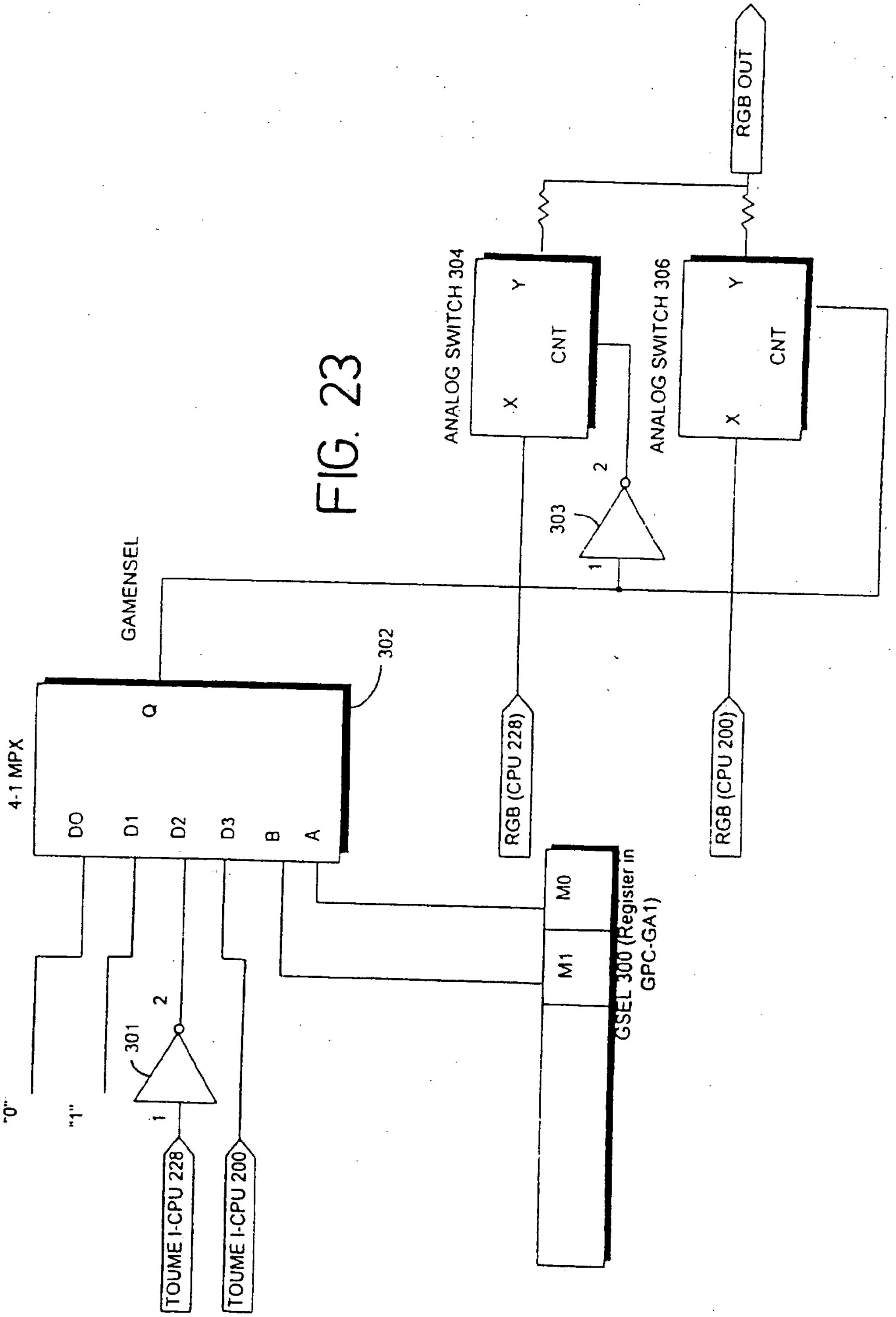

4-1 MPX
DO
D1
D2
D3
B
A
Q
302
GAMENSEL
"0"
"1"
301
1
2
TOUME I-CPU 228
TOUME I-CPU 200
M0
M1
GSEL 300 (Register in GPC-GA1)
RGB (CPU 228)
RGB (CPU 200)
303
1
2
ANALOG SWITCH 304
ANALOG SWITCH 306
X
Y
CNT
X
Y
CNT
RGB OUT

FIG. 24B

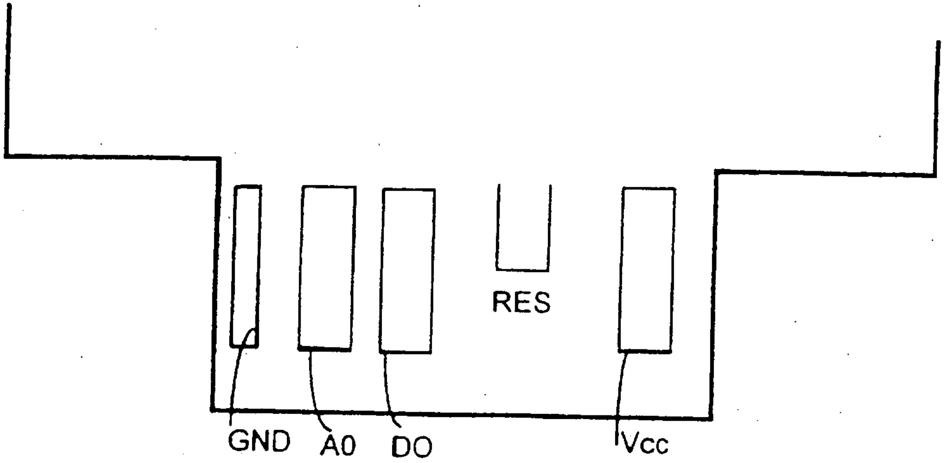
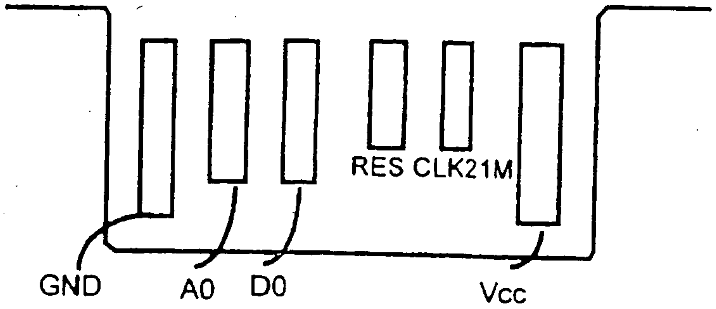


FIG. 24D



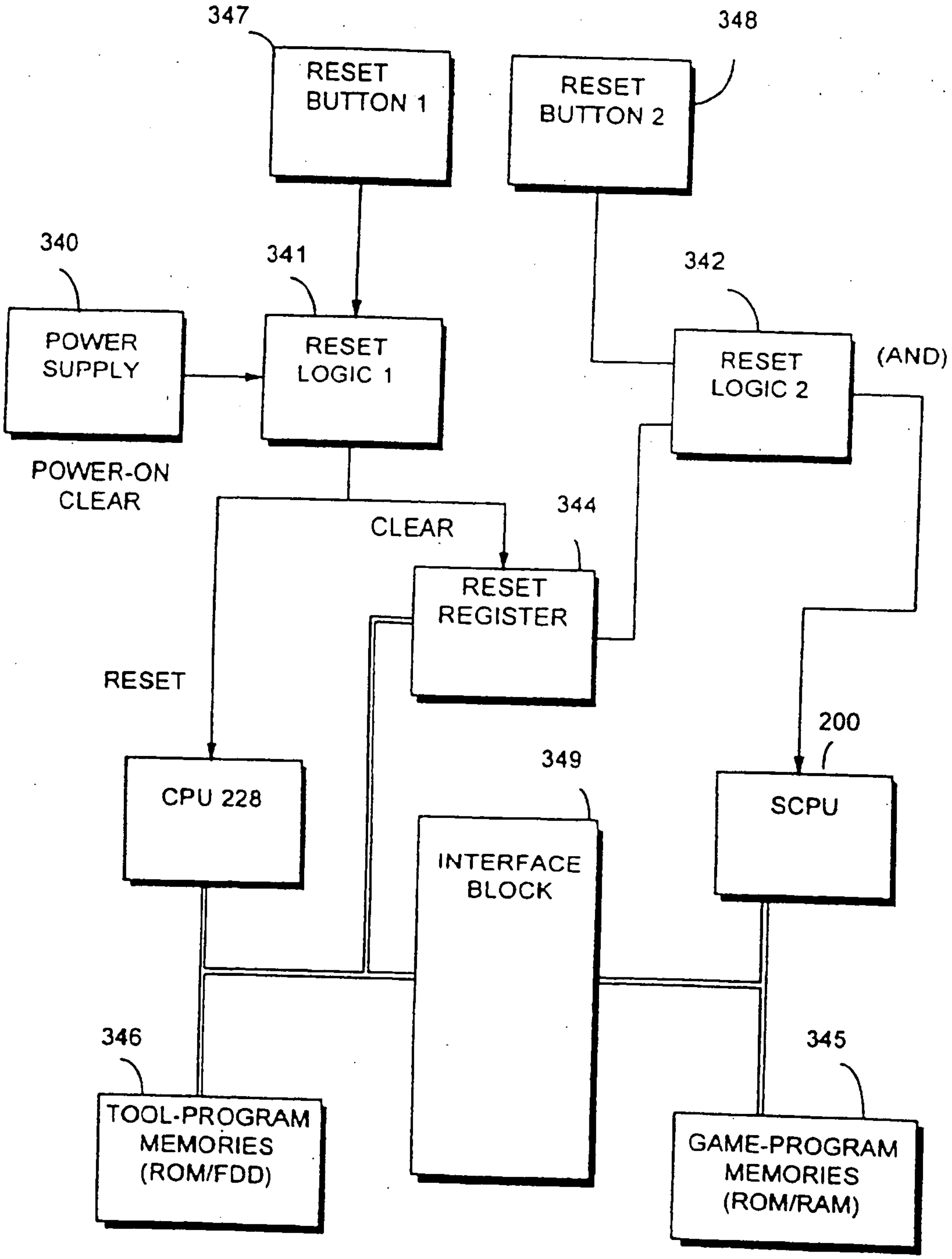

347
RESET BUTTON 1
348
RESET BUTTON 2
340
341
342
POWER SUPPLY
RESET LOGIC 1
RESET LOGIC 2
(AND)
POWER-ON CLEAR
CLEAR
344
RESET REGISTER
RESET
200
349
CPU 228
SCPU
INTERFACE BLOCK
346
345
TOOL-PROGRAM MEMORIES (ROM/FDD)
GAME-PROGRAM MEMORIES (ROM/RAM)

FIG. 25

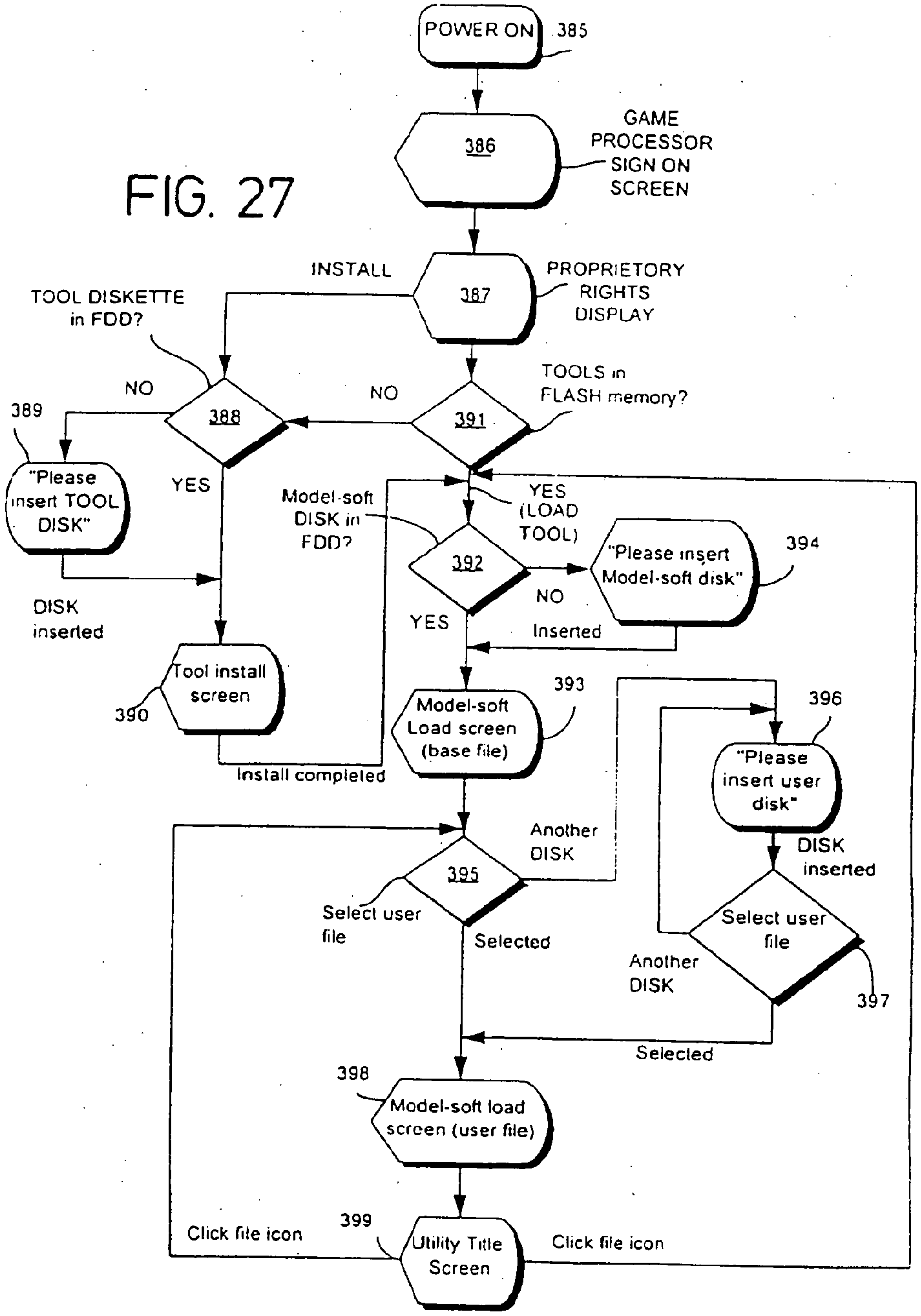

POWER ON
385
386
GAME PROCESSOR SIGN ON SCREEN
INSTALL
387
PROPRIETORY RIGHTS DISPLAY
TOOL DISKETTE in FDD?
NO
389
388
NO
391
TOOLS in FLASH memory?
"Please insert TOOL DISK"
YES
Model-soft DISK in FDD?
YES (LOAD TOOL)
392
"Please insert Model-soft disk"
394
NO
DISK inserted
YES
Inserted
Tool install screen
390
Install completed
393
Model-soft Load screen (base file)
396
"Please insert user disk"
Another DISK
395
DISK inserted
Select user file
Selected
Select user file
Another DISK
397
Selected
398
Model-soft load screen (user file)
399
Click file icon
Utility Title Screen
Click file icon

FIG. 27

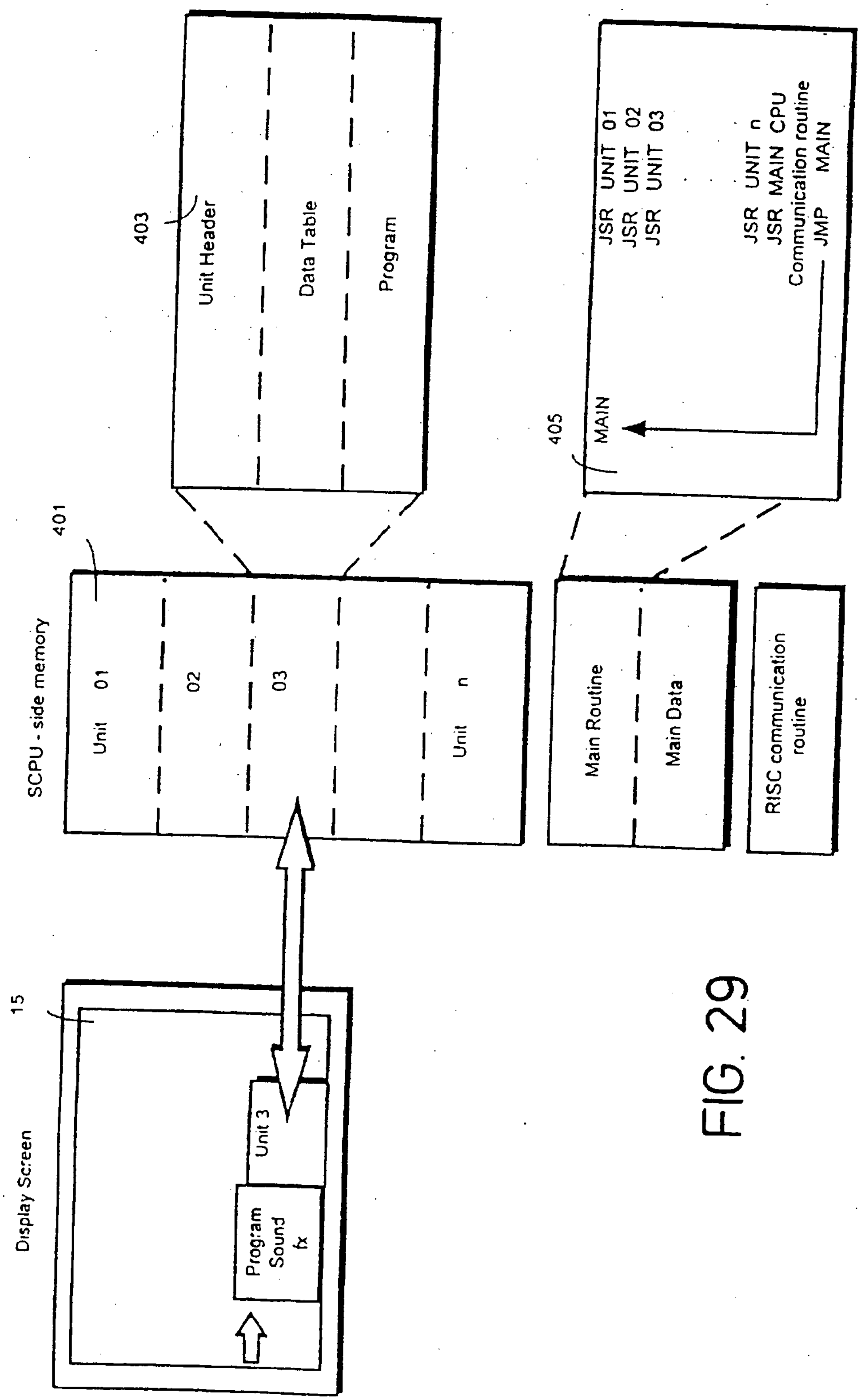

Display Screen
15
Program
Sound
fx
Unit 3
SCPU - side memory
401
Unit 01
02
03
Unit n
403
Unit Header
Data Table
Program
Main Routine
Main Data
RISC communication
routine
405
MAIN
JSR UNIT 01
JSR UNIT 02
JSR UNIT 03
JSR UNIT n
JSR MAIN CPU
Communication routine
JMP MAIN

FIG. 29

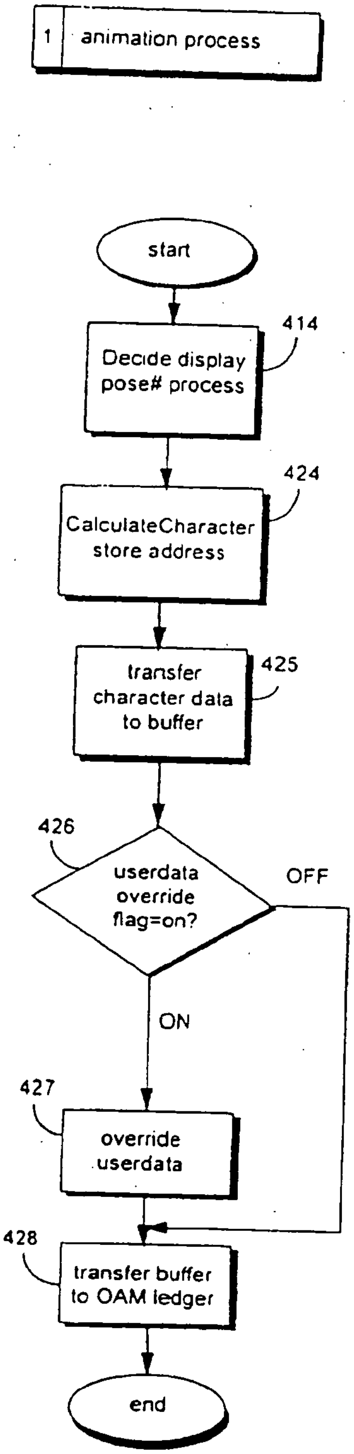

1
animation process
start
Decide display pose# process
414
CalculateCharacter store address
424
transfer character data to buffer
425
426
userdata override flag=on?
OFF
ON
427
override userdata
428
transfer buffer to OAM ledger
end

FIG. 30B

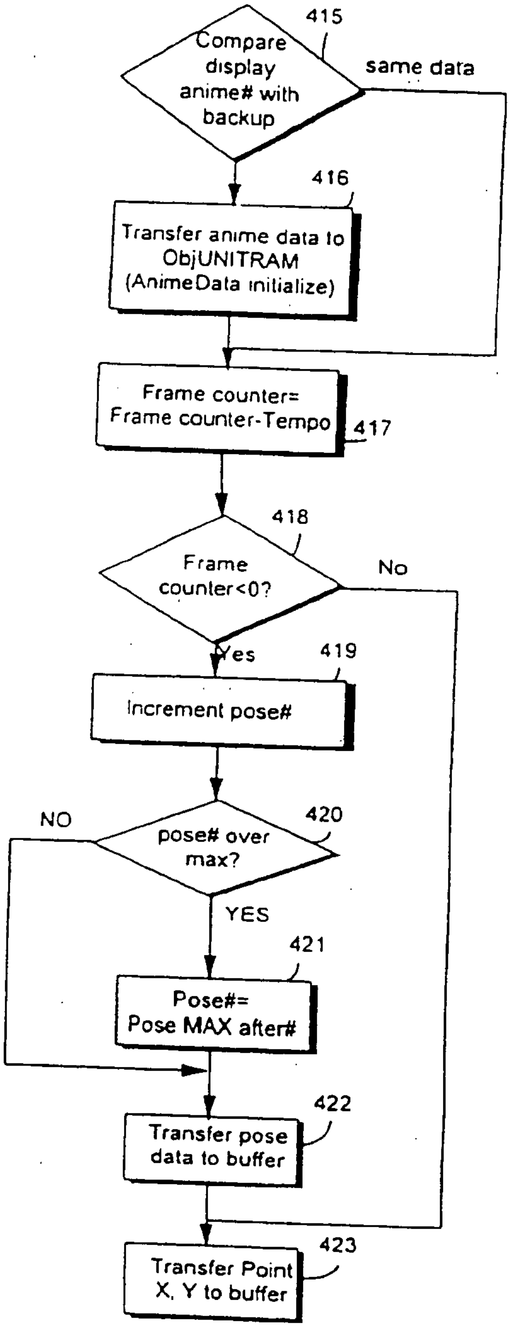

415
Compare display anime# with backup
same data
416
Transfer anime data to ObjUNITRAM (AnimeData initialize)
Frame counter= Frame counter-Tempo
417
418
Frame counter<0?
No
Yes
419
Increment pose#
NO
420
pose# over max?
YES
421
Pose#= Pose MAX after#
422
Transfer pose data to buffer
423
Transfer Point X, Y to buffer

FIG. 30C

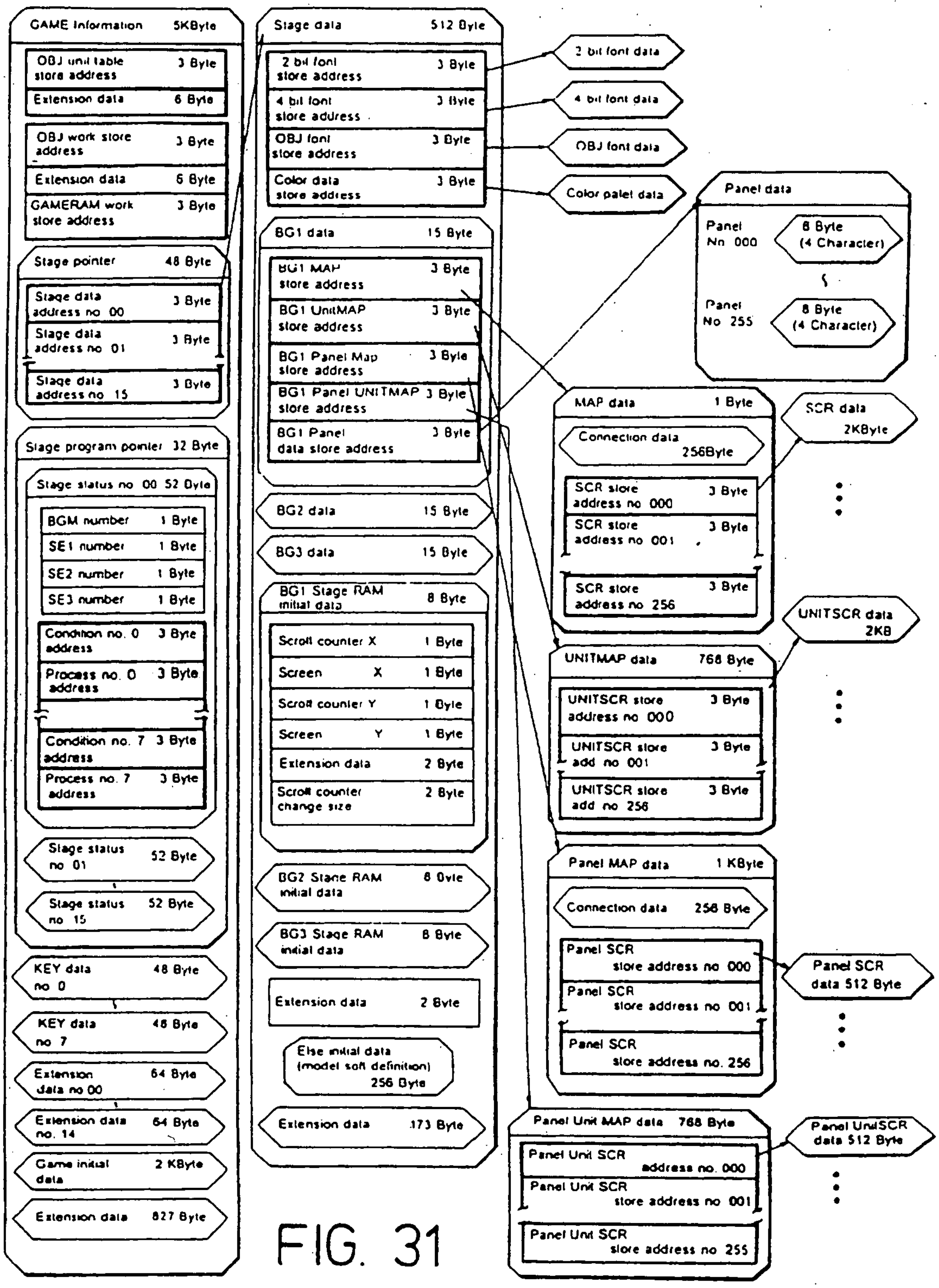

GAME Information 5KByte
OBJ unit table store address 3 Byte
Extension data 6 Byte
OBJ work store address 3 Byte
Extension data 6 Byte
GAMERAM work store address 3 Byte
Stage pointer 48 Byte
Stage data address no 00 3 Byte
Stage data address no 01 3 Byte
Stage data address no 15 3 Byte
Stage program pointer 32 Byte
Stage status no 00 52 Byte
BGM number 1 Byte
SE1 number 1 Byte
SE2 number 1 Byte
SE3 number 1 Byte
Condition no. 0 address 3 Byte
Process no. 0 address 3 Byte
Condition no. 7 address 3 Byte
Process no. 7 address 3 Byte
Stage status no 01 52 Byte
Stage status no 15 52 Byte
KEY data no 0 48 Byte
KEY data no 7 48 Byte
Extension data no 00 64 Byte
Extension data no. 14 64 Byte
Game initial data 2 KByte
Extension data 827 Byte
Stage data 512 Byte
2 bit font store address 3 Byte
4 bit font store address 3 Byte
OBJ font store address 3 Byte
Color data store address 3 Byte
BG1 data 15 Byte
BG1 MAP store address 3 Byte
BG1 UnitMAP store address 3 Byte
BG1 Panel Map store address 3 Byte
BG1 Panel UNITMAP store address 3 Byte
BG1 Panel data store address 3 Byte
BG2 data 15 Byte
BG3 data 15 Byte
BG1 Stage RAM initial data 8 Byte
Scroll counter X 1 Byte
Screen X 1 Byte
Scroll counter Y 1 Byte
Screen Y 1 Byte
Extension data 2 Byte
Scroll counter change size 2 Byte
BG2 Stage RAM initial data 8 Byte
BG3 Stage RAM initial data 8 Byte
Extension data 2 Byte
Else initial data (model soft definition) 256 Byte
Extension data 173 Byte
2 bit font data
4 bit font data
OBJ font data
Color palet data
Panel data
Panel No 000
8 Byte (4 Character)
Panel No 255
8 Byte (4 Character)
MAP data 1 Byte
Connection data 256Byte
SCR store address no 000 3 Byte
SCR store address no 001 3 Byte
SCR store address no 256 3 Byte
SCR data 2KByte
UNITMAP data 768 Byte
UNITSCR store address no 000 3 Byte
UNITSCR store add no 001 3 Byte
UNITSCR store add no 256 3 Byte
UNITSCR data 2KB
Panel MAP data 1 KByte
Connection data 256 Byte
Panel SCR store address no 000
Panel SCR store address no 001
Panel SCR store address no. 256
Panel SCR data 512 Byte
Panel Unit MAP data 768 Byte
Panel Unit SCR address no. 000
Panel Unit SCR store address no 001
Panel Unit SCR store address no 255
Panel UnitSCR data 512 Byte

FIG. 31

GAME RAM 3 K BYTE

EXTENSION		1 BYTE
STAGE NUMBER		1 BYTE
STATUS REGISTER		1 BYTE
STATUS REGISTER BACKUP		1 BYTE
BG 1		1 2 BYTE
	X POINT	3 BYTE
	Y POINT	3 BYTE
	X SPEED	2 BYTE
	Y SPEED	2 BYTE
	EXTENSION	2 BYTE
BG 2		1 2 BYTE
BG 3		1 2 BYTE
EXTENTION		3 BYTE
STAGE DATA		3 BYTE
GAME DATA		2 K BYTE
EXTENTION		980 BYTE

FIG. 33

OBJ UNIT RAM 128 BYTE

UNIT ID		1 BYTE
UNIT ID (BACKUP)		1 BYTE
STATUS REGISTER		1 BYTE
STATUS REGISTER BACKUP		1 BYTE
POINT X		3 BYTE
Y		3 BYTE
EXTENSION		3 BYTE
OBJ FORMAT		1 BYTE
CHARACTER SIZE		1 BYTE
POSE MAX		1 BYTE
POSE MAX AFTER NUMBER		1 BYTE
OBJ SIZE		1 BYTE
HIT SIZE	X	1 BYTE
	Y	1 BYTE
TEMPO	LOW	1 BYTE
	HIGH	1 BYTE
ATTRIBUTE		1 BYTE
CHARACTER STORE ADDRESS		3 BYTE
FRAME COUNTER	LOW	1 BYTE
	HIGH	1 BYTE
DISPLAY ANIME NUMBER		1 BYTE
DISPLAY ANIME NUMBER (BACKUP)		1 BYTE
DISPLAY POSE NUMBER		1 BYTE
DISPLAY POSE NUMBER (BACKUP)		1 BYTE
USER ARIA	FLAG	1 BYTE
	FLIP	1 BYTE
	PRIORITY	1 BYTE
	ATTRIBUTE	1 BYTE
ANIME COUNTER		1 BYTE
AUTO PROGRAM ARIA	COUNTER	1 BYTE
	TEMPO	1 BYTE
	WORK	1 BYTE
	STORE ADDRESS	3 BYTE
INITIAL DATA		6 4 BYTE
EXTENTION		2 1 BYTE

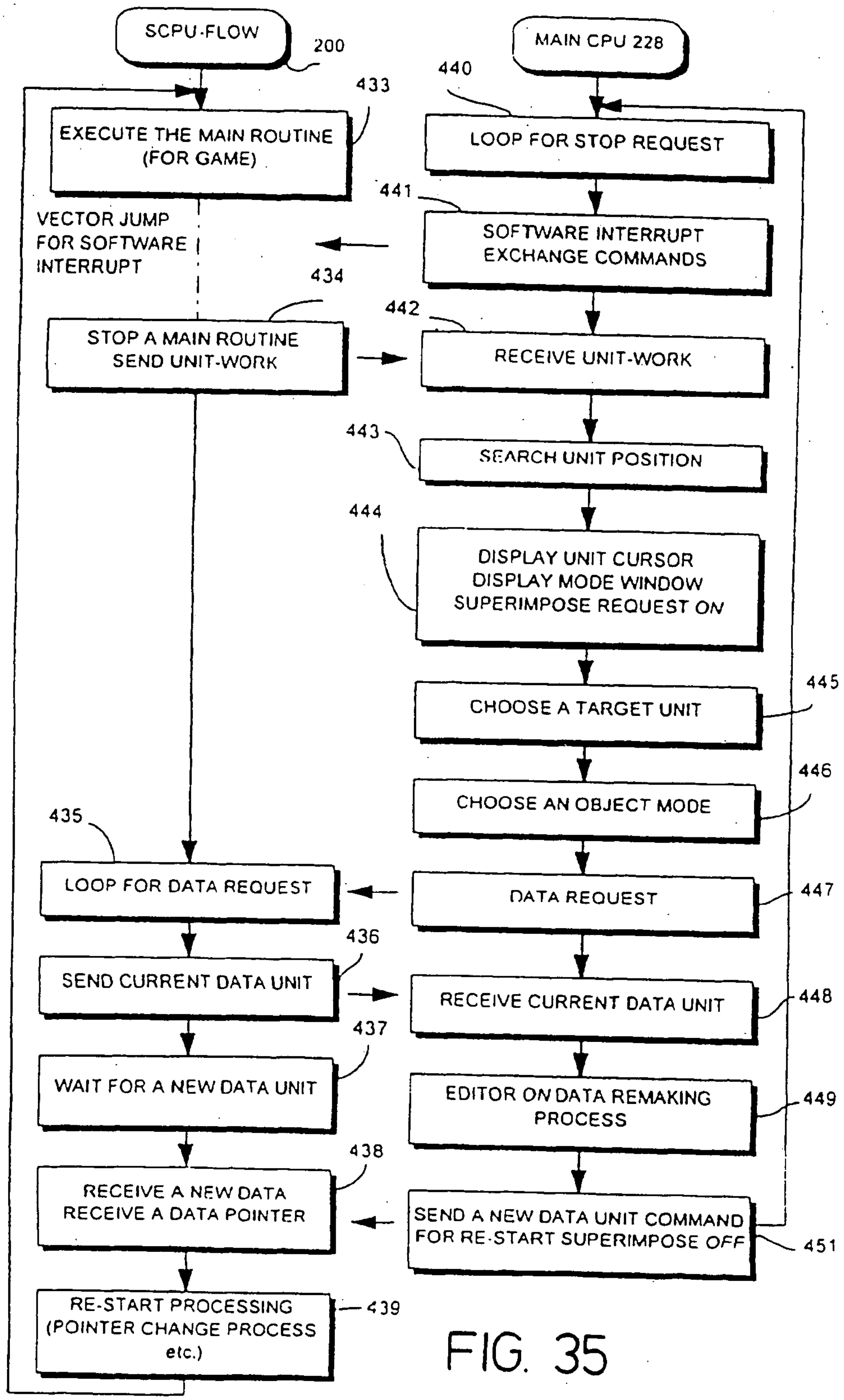

SCPU-FLOW
200
MAIN CPU 228
440
433
EXECUTE THE MAIN ROUTINE (FOR GAME)
LOOP FOR STOP REQUEST
441
VECTOR JUMP FOR SOFTWARE INTERRUPT
SOFTWARE INTERRUPT EXCHANGE COMMANDS
434
442
STOP A MAIN ROUTINE SEND UNIT-WORK
RECEIVE UNIT-WORK
443
SEARCH UNIT POSITION
444
DISPLAY UNIT CURSOR DISPLAY MODE WINDOW SUPERIMPOSE REQUEST ON
445
CHOOSE A TARGET UNIT
446
CHOOSE AN OBJECT MODE
435
LOOP FOR DATA REQUEST
DATA REQUEST
447
436
SEND CURRENT DATA UNIT
RECEIVE CURRENT DATA UNIT
448
437
WAIT FOR A NEW DATA UNIT
EDITOR ON DATA REMAKING PROCESS
449
438
RECEIVE A NEW DATA RECEIVE A DATA POINTER
SEND A NEW DATA UNIT COMMAND FOR RE-START SUPERIMPOSE OFF
451
RE-START PROCESSING (POINTER CHANGE PROCESS etc.)
439

FIG. 35

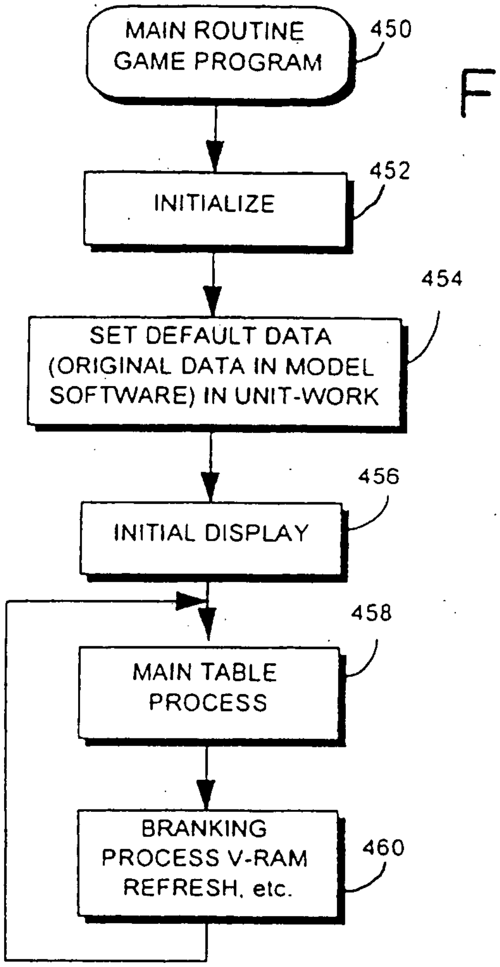

MAIN ROUTINE GAME PROGRAM
450
INITIALIZE
452
SET DEFAULT DATA (ORIGINAL DATA IN MODEL SOFTWARE) IN UNIT-WORK
454
INITIAL DISPLAY
456
MAIN TABLE PROCESS
458
BRANKING PROCESS V-RAM REFRESH, etc.
460

FIG. 38A

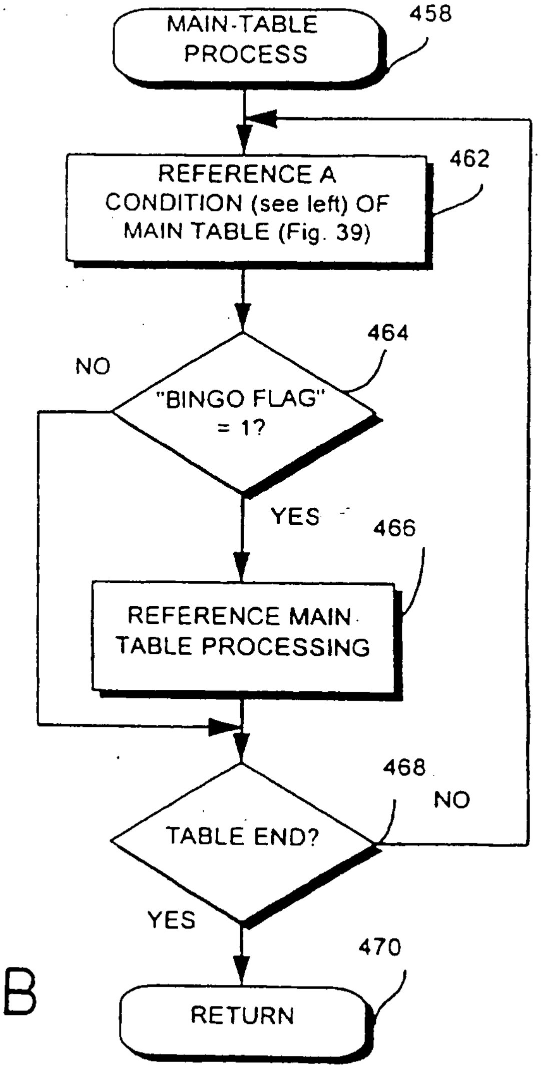

MAIN-TABLE PROCESS
458
REFERENCE A CONDITION (see left) OF MAIN TABLE (Fig. 39)
462
"BINGO FLAG" = 1?
464
NO
YES
REFERENCE MAIN TABLE PROCESSING
466
TABLE END?
468
NO
YES
RETURN
470

FIG. 38B

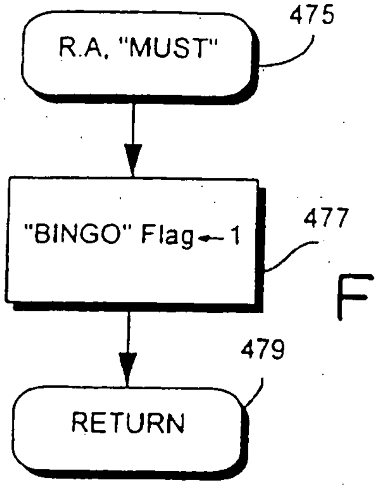

R.A. "MUST"
475
"BINGO" Flag←1
477
479
RETURN

FIG. 40

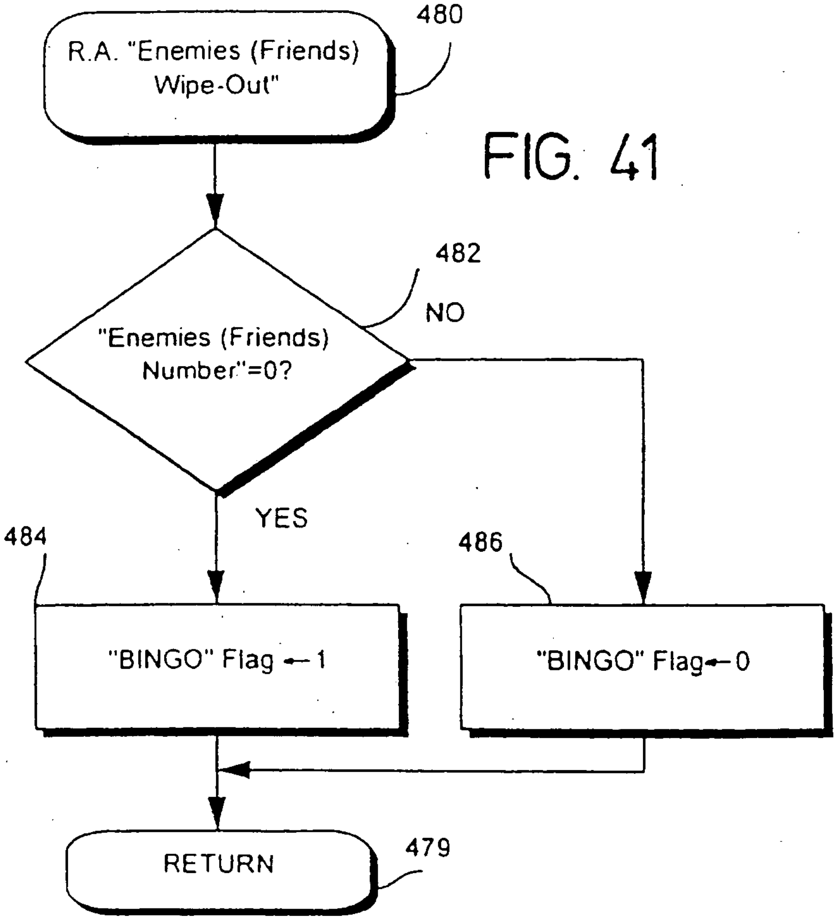

R.A. "Enemies (Friends) Wipe-Out"
480
482
NO
"Enemies (Friends) Number"=0?
YES
484
486
"BINGO" Flag ←1
"BINGO" Flag←0
RETURN
479

FIG. 41

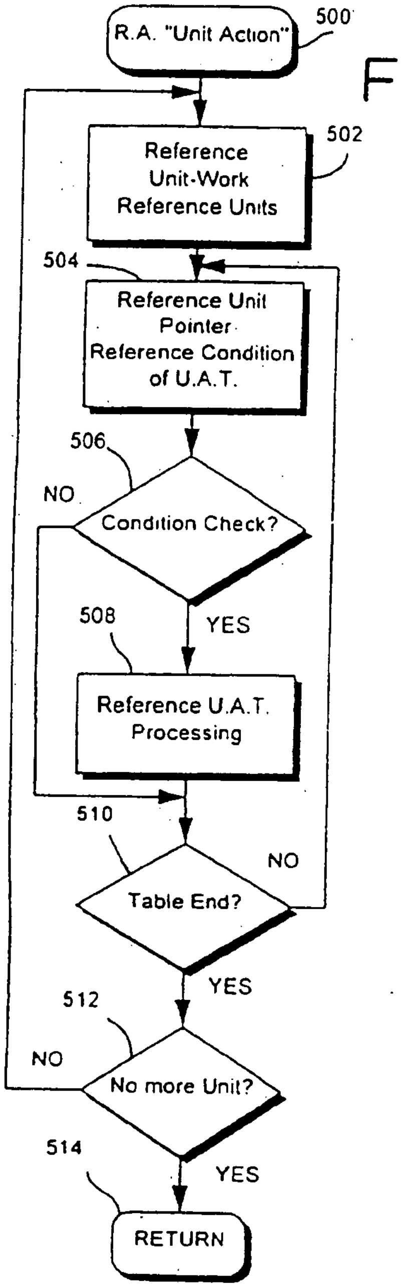

R.A. "Unit Action"
500
502
Reference Unit-Work Reference Units
504
Reference Unit Pointer Reference Condition of U.A.T.
506
NO
Condition Check?
YES
508
Reference U.A.T. Processing
510
Table End?
NO
YES
512
NO
No more Unit?
YES
514
RETURN

FIG. 43

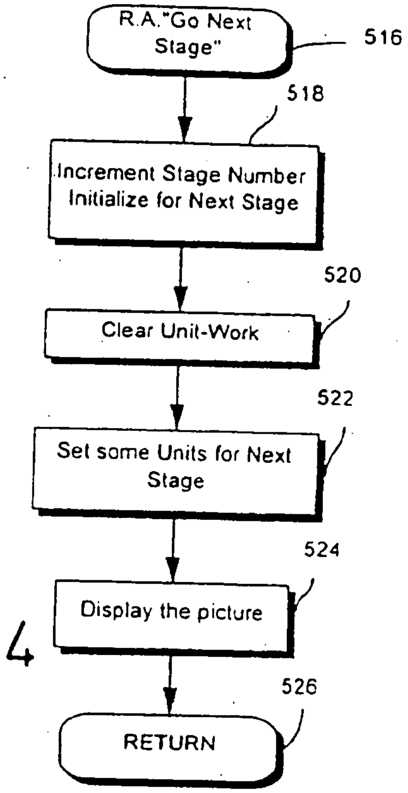

R.A."Go Next Stage"
516
518
Increment Stage Number Initialize for Next Stage
520
Clear Unit-Work
522
Set some Units for Next Stage
524
Display the picture
526
RETURN

FIG. 44

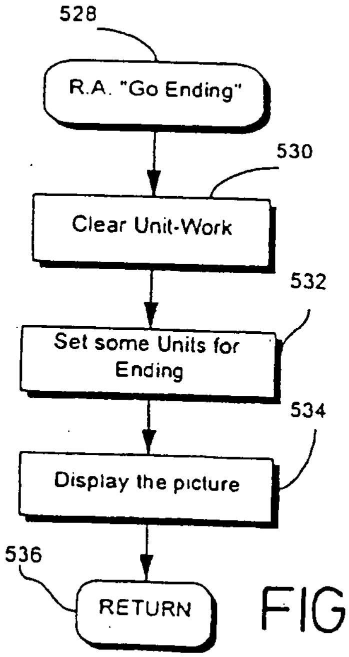


FIG. 45

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 30 7727

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
D,A	US-A-4 799 635 (NAKAGAWA) 24 January 1989 * the whole document * ---	1,2,4,5,9	G06F9/44 A63F9/22
A	GB-A-2 033 703 (SANDERS ASSOCIATES, INC.) 21 May 1980 * the whole document * ---	1,5,9	
A	EP-A-0 597 316 (VIRTUAL PROTOTYPES, INC.) 18 May 1994 * page 2, line 11 - page 7, line 16 * ---	1,5,9	
A	JONATHAN MENDOZA: 'The Official DOOM Survivor's Strategies and Secrets' 1994 , SYBEX , SAN FRANCISCO, US * page 281, line 13 - page 286, line 19 * -----	9,17	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	14 February 1996	Fonderson, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

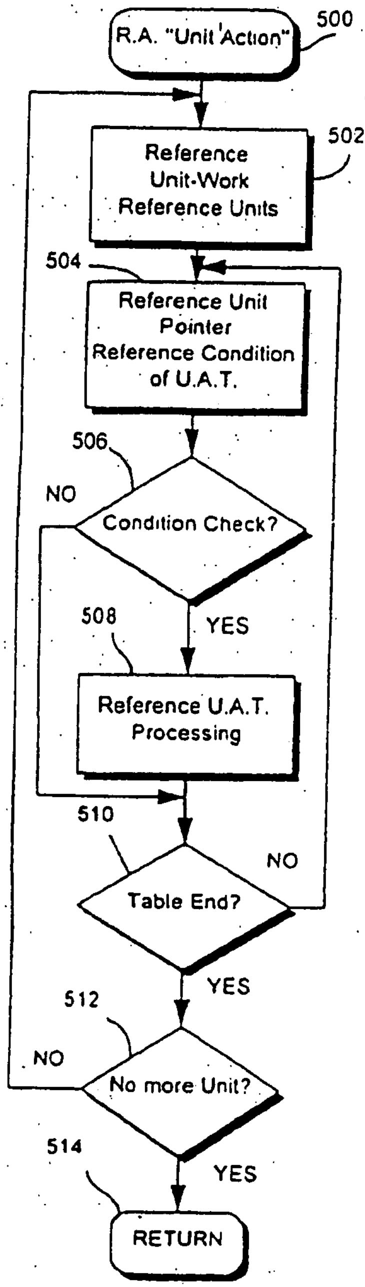

R.A. "Unit Action"
500
Reference Unit-Work Reference Units
502
504
Reference Unit Pointer Reference Condition of U.A.T.
506
NO
Condition Check?
YES
508
Reference U.A.T. Processing
510
Table End?
NO
YES
512
NO
No more Unit?
YES
514
RETURN

FIG. 43

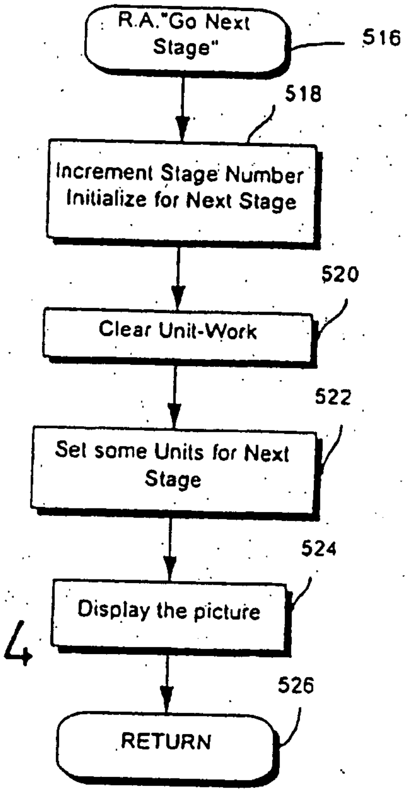

R.A."Go Next Stage"
516
518
Increment Stage Number Initialize for Next Stage
520
Clear Unit-Work
522
Set some Units for Next Stage
524
Display the picture
526
RETURN

FIG. 44

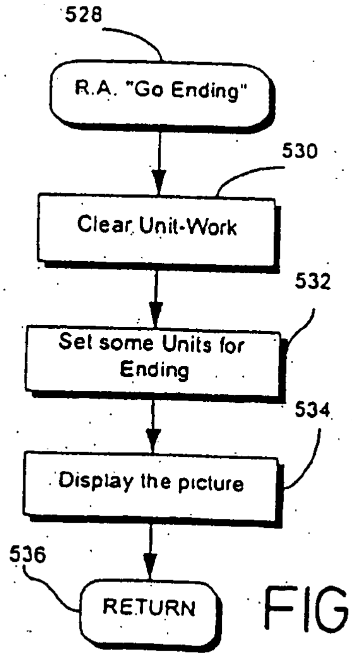


FIG. 45